Lightning Talks of EduHPC 2022

Apan Qasem[1], Hartwig Anzt[2,3], Eduard Ayguade[4], Katharine Cahill[5], Ramon Canal[4], Jany Chan[6], Eric Fosler-Lussier[6], Fritz Göbel[2], Arpan Jain[6], Marcel Koch[2], Mateusz Kuzak[7], Josep Llosa[4], Raghu Machiraju[6], Xavier Martorell[4], Pratik Nayak[2], Shameema Oottikkal[5], Marcin Ostasz[8], Dhabaleswar K. Panda[6], Dirk Pleiter[9], Rajiv Ramnath[6] Maria-Ribera Sancho[4], Alessio Sclocco[7], Aamir Shafi[6], Hanno Spreeuw[7], Hari Subramoni[6], Karen Tomko[7]

[1]Department of Computer Science, Texas State University, USA
[2]Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
[3]University of Tennessee (UTK), Knoxville, USA
[4]Barcelona Supercomputing Center and Universitat Politècnica de Catalunya, Spain
[5]Ohio Supercomputer Center, USA
[6]College of Engineering, The Ohio State University, USA
[7]Netherlands eScience Center, The Netherlands
[8]ETP4HPC, The Netherlands
[9]PDC Center for High Performance Computing, KTH Royal Institute of Technology, Sweden

***Abstract*—The lightning talks at EduHPC provide an opportunity to share early results and insights on parallel and distributed computing (PDC) education and training efforts. The four lightning talks at EduHPC 2022 cover a range of topics in broadening PDC education: (i) curriculum development efforts for the European Masters in HPC program, (ii) bootcamps for CI professionals who support the running of AI workloads on HPC systems, (iii) a GPU programming course following the Carpentries model and (iv) peer-review assignments to help students write efficient parallel algorithms within sustainable software libraries.**

***Index Terms*—computer science education, high performance computing education, Masters curriculum, CI professional training, distance learning, peer review assignments, GPU computing**

I. Introduction

The breakdown of Dennard Scaling coupled with the need for increased performance per watt have ushered in the era of ubiquitous parallel computing. Integration of high-performance CPUs with GPUs is now common in all classes of HPC systems. Heterogeneous parallel systems have also permeated other computing domains such as mobile processing, cloud computing and the Internet of Things (IoTs). Concomitant to these architectural changes, the workloads for HPC systems have become more diverse. ML practitioners routinely use accelerated computing and a wide range of applications are being developed with Cloud-HPC in mind.

Given this state of pervasive parallelism, it is imperative that computing students develop a broad understanding of parallel, heterogeneous and distributed computing so that they are equipped with the requisite skills to program these complex systems. Furthermore, considering that the educational context and the needs of students can vary across institutions, it is important to develop and explore a wide range of approaches for teaching PDC. EduHPC, the Workshop on Education for High-Performance Computing, provides a forum to share such efforts on improving PDC education. The lightning talks at EduHPC specifically focuses on new and innovative approaches to teaching PDC that may not have been fully tested in the field. This paper presents an overview of the four lightning talks at EduHPC22 which cover a range of topics:

- **European Masters in HPC:** Recently, the EUMaster4HPC project has been started with the ambition of boosting the education of HPC experts at universities throughout Europe. Within this project, a future European curriculum for a Master in HPC is being developed. This talk reports on initial efforts towards establishing a set of necessary skills.
- **AI Bootcamps:** A team at the Ohio Supercomputer Center and the Ohio State University is developing a series of bootcamps for Cyberinfrastructure (CI) Professionals to increase support expertise for researchers with Artificial Intelligence (AI) workloads running at research computing facilities. The talk shares experiences from completing the first six-week virtual program of core foundations topics in AI.
- **GPU Carpentry:** GPUs are nowadays used to accelerate applications in multiple scientific domains, and is therefore necessary even for researchers outside of computer science to learn how to use them. However, traditional GPU programming courses are often aimed at people with a computer science or HPC background. This talk presents an open-source GPU programming course, following the Carpentries pedagogical style which makes it accessible to non computer scientists.
- **Peer-review Assignments** A peer review based assignment approach that teaches students to write efficient parallel algorithms for different hardware architectures by providing a build, test continuous integration framework. The aim is to train a new generation of research software engineers (RSE) who will incorporate sustainable software development practices into their software libraries

and their research software in general.

Section II describes the European Masters talk, Section III describes the AI bootcamp talk. Section IV describes the GPU Carpentries talk and finally Section V describes the peer-review assignments talk.

II. Towards a European Curriculum for a Master in HPC

By: Dirk Pleiter, Maria-Ribera Sancho, Xavier Martorell, Josep Llosa, Ramon Canal, Eduard Ayguade, Marcin Ostasz

In 2018, EuroHPC [1] was established as an organization for leading the European supercomputing efforts. Through this organization, the European Commission, as well as the EuroHPC participating states, are massively investing in an HPC-based infrastructure, as well as the development of HPC-related technologies. A major challenge to this effort being successful is the availability of well-trained experts in the area of HPC. For this reason, EuroHPC is funding the EUMaster4HPC project [2] that has, in particular, the goal of developing a new and innovative European Master program for HPC.

This program involves both, the education of two cohorts of Master students involving 8 universities distributed all over Europe as well as the development of a European curriculum. The latter involves a large number of partners including additional universities, supercomputing centers, and industrial organizations.

In this context, the following strategy has been adopted: Starting-point is the documentation and analysis of requirements of the future European labor market for HPC experts through a list of skills. This list of skills can be used as a checklist when creating the framework and structure of a European Master program, defining its learning outcomes as well as the academic content and modules for both, fundamentals and specializations.

In this short paper, we report on the methods adopted for assessing the requirements starting with the description of the methodology in section II-A. Thereafter, initial results will be presented in section II-B before providing an outlook on the next steps in section II-C.

A. Methodology

In a first step towards the creation of a list of necessary skills, job offers for positions in the academic and non-academic sectors have been collected separately.

In a second step, the job advertisements were analyzed to identify a set of job profiles and for each job profile a set of skills. One challenge of this approach is that job profiles are not standardized and different employers follow different definitions and interpretations. Sufficient similarity of different job profiles could be identified such that they can be understood as classes of jobs with profiles that are sufficiently distinguishable.

This information was the basis for the next steps. For the non-academic sector consisted of collecting input from stakeholders through online questionnaires as well as structured interviews. The questionnaire and the input for the interviews were based on the identified job profiles and associated set of skills. The interview partners were encouraged to consider the input as a starting point for dialogue and to express their views on which job profiles and which skills they considered to be important. The purpose was also to reduce a possible bias introduced by the initially chosen set of job advertisements.

In the academic sector, the initial study was discussed in an internal workshop. The partners provided their input on the classification, areas, and skills found in the initial study. In particular, the participants were asked to check that (1) there were no missing profiles, (2) the profiles included all the hard skills in their area of expertise, and (3) the list of soft skills was complete.

Finally, the stakeholder feedback was used for compiling a resulting list of skills.

B. Initial Results

For this initial study, 135 job offers from the academic sector, as well as almost 450 job offers published by Barcelona Supercomputing Center (one of the largest research centers in HPC technologies in Europe) during the last 5 years have been collected. For the non-academic sector, about 30 advertisements have been gathered that mainly concerned positions based in Europe. The main sources had been HPCWire[1] and HiPEACJobs[2] in Spring 2022.

From the advertisements in the academic sector, the following job profiles had been identified (in brackets the fraction of advertisements is shown): Applications (30%), parallel programming and tool support (18%), DevOps (31%), manager/consultant (0.5%), and system architect (21%).

The evaluation of the job offers from the academic sector performed in the internal workshop, resulted in the following set of profiles: Application / Domain Expert, Parallel Programming and Tools Support/Solution Designer, DevOps (System Support and Development), and System Architect. Regarding the Manager/Consultant profile, it was decided not to include the profile in the Master curricula as those job offers ask for senior-level experience and this is not the target group of this Master. For all the proposed profiles, the needed hard skills and the common soft skills were identified.

The evaluation of the job offers from the non-academic sector resulted in a slightly different set of job profiles: Computational scientist, hardware developer, system software developer, HPC architect, system administrator, and application software developer. In total 6 interviews with a duration of 1-2 hours were conducted. Additionally, 12 replies to an online questionnaire had been collected. For these interview with non-academic stakeholders we aimed for high diversity. We had interview partners targeting different market segments, e.g. a semiconductor company, HPC technology solution and platform providers, HPC integrators, and companies providing

[1] https://jobs.hpcwire.com

[2] https://www.hipeac.net/jobs

products and services in the area of aircraft and oil&gas market. Furthermore, both large-scale European and international companies as well as SMEs have been interviewed.

The interview partners from the non-academic sector confirmed the identified list of job profile classes as being the most relevant. It was, however, also noted that new job profiles may become relevant. One example is the possible job profile of a data architecture specialist, who is expected to come with knowledge and experience in data-centric architectures that may be optimized for high-throughput computing applications. Also in the context of industrial research and development efforts related to the design of digital twins will require additional skills like the ability to use AI models or expertise in data extraction and visualization.

Several interview partners stressed the importance of a broad skill set, which suggests that it will be important for the planned curriculum to comprise a large set of skills adaptable to many job profiles. A recurring topic during the interviews was both, the understanding of modern hardware, the relevant architectures and technologies as well as the ability to exploit the underlying hardware resources efficiently. This means that good knowledge and understanding of computer architectures must be foreseen as an important part of the Master in HPC education.

The interview partners also highlighted that they consider a good understanding of the basic principles more important than a detailed knowledge of certain technologies. The rationale is that most architectures and technologies are based on principles that stay relevant for longer periods of time, like the principles of a von Neumann processor architecture that continue to apply to all current HPC processors as well as compute accelerators like GPUs.

One outcome of the interviews was the reaffirmation of the importance of soft skills. The importance of soft skills in IT industry has already been identified more than 10 years ago (see, e.g., [3] and references therein). Different ways of training soft skills in the context of computer science education have been proposed (see, e.g., [4], [5]). But none of these are widely implemented as education typically focuses on technical skills. In the context of our efforts the need for soft skills training was raised by several of the interviewed companies. While there is overlap with earlier formulated requirements (see, e.g., [3]) the feedback contains interesting nuances. Interviewees suggested focusing on developing high levels of curiosity, encouraging a willingness to ask questions and relying on other people's expertise, and encouraging ambition to acquire more skills and knowledge.

C. Outlook

During an early phase of the EUMaster4HPC project, which has the ambition of formulating a curriculum for a European Master in HPC, an extensive list of skills has been collected that reflect the requirements of the HPC-related labor market. The project is now starting the work on the curriculum. The list of skills will serve as a checklist to guide this effort and ensure that students will be trained taking into account the needs of future employers. This approach will allow to highlight highly interesting career opportunities, which can be used for attracting more students to pursue a Master in HPC degree.

Acknowledgments

The authors would like to thank the various organizations for their significant efforts to provide input through questionnaires and interviews. Funding for this work has been received from the EuroHPC Joint Undertaking under Grant Agreement 101051997 (EUMaster4HPC).

III. An Artificial Intelligence Bootcamp for Cyberinfrastructure Professionals

By: Katharine Cahill, Jany Chan, Eric Fosler-Lussier, Arpan Jain, Raghu Machiraju, Shameema Oottikkal, Dhabaleswar K. Panda, Rajiv Ramnath, Aamir Shafi, Hari Subramoni, Karen Tomko

Artificial Intelligence (AI) is used in many aspects of modern life such as language translation and image analysis. In addition to consumer and business applications, researchers are increasingly using AI techniques in their scientific processes. The growth in AI is heavily dependent on new Deep Learning (DL) and Machine Learning (ML) schemes. As datasets and DL and ML models become more complex the computing requirements for AI increase and researchers turn to high performance computing (HPC) facilities to meet these needs. This is leading to a critical need for a Cyberinfrastructure (CI) workforce that supports HPC systems with expertise in AI techniques and underlying technology.

We are piloting a bootcamp-style training model to address this AI workforce gap. Our premise, after attending the bootcamp CI professionals are better equipped to provide computing and data services to AI research users. This in turn will broaden adoption and effective use of advanced CI by researchers in a wide range of disciplines and will leave an impact on science and corresponding benefits to society from their successes.

In this lightning talk we will give an overview of our CyberTraining project to pilot the bootcamps, our learning objectives and experiences from our inaugural offerings.

A. Description of Work

Our AI Bootcamps for CI professionals have the overarching goal of increasing the confidence and effectiveness of their support of AI researchers. We leverage the CI professionalization efforts of the Campus Research Computing Consortium (CaRCC) to organize our training outcomes based on four career facings [6]: Strategy/Policy facing, Researcher facing, Software/Data facing, and Systems facing. We can identify learning outcomes for each CI facing and organize training tracks customized to specific roles. For this pilot we are focused on developing a comprehensive training experience for Software/Data facing CI professionals. The AI Bootcamp is offered virtually in two sections of six weeks each. Taking advantage of tools used in the CS classroom as well as those developed at OSC for access to research computing

resources, this program combines lecture, discussion, and hands-on activities to engage participants.

The novelty of this project is its holistic approach to addressing the AI expertise gap for CI professionals. Our project team is comprised of CI professionals, experienced in training CI users and providing CI operations, and Computer Science faculty members, experienced in offering courses in Data Analytics, AI, and High-Performance AI with active AI-based research programs. Drawing on extensive experience and materials in hands-on experiential learning for AI, the team is continuing to put together a comprehensive curriculum spanning foundational AI, software frameworks, and high-performance computing for AI in a modularized virtual format to minimize barriers to access for the CI professional learner.

We have identified learning outcomes and developed corresponding curricula for the AI Bootcamp Core foundations track as a pre-requisite track for all CI Professionals. The curricula included the following topics: Python tools for data Analysis and typical data types (tables, images, time series, maps and text); Fundamentals of Machine Learning, Bayesian Modeling and Neural Networks; Machine Learning and Deep Learning software frameworks; Parallel and distributed DNN training; Data science using Dask; Challenges in exploiting HPC technologies for DL, ML, and data science.

B. Outcomes

We carried out a six-week AI Bootcamp covering the Core Foundations Track (CFT) from March 22nd to April 27th, 2022. The bootcamp was offered two days per week for two hours virtually via Zoom. The instruction included a mix of lectures, questions and answers, and exercises. Exercises progressed from notebooks provided via web interfaces to command line exercises submitted via traditional HPC batch submission at Ohio Supercomputer Center. We have utilized classroom services offered by Ohio Supercomputer Center and Google Colab.

There was strong interest from the community in the bootcamp. We had 156 responses to our Call for Participation (CFP) within 48 hours of posting. We invited 62 professionals to attend the initial bootcamp. These participants represented 50 organizations from 28 states. We are getting ready for our Fall 2022 AI Bootcamp covering the Software and Data Facing Track (SDFT). It is being offered to participants of the spring CFT bootcamp. More than half of the spring attendees are planning to return. The SDFT bootcamp will provide a deeper dive into the subjects visited last spring by walking through the steps in a typical AI pipeline. For each step in the pipeline, we will cover best practices and state-of-the-art software and tools. We will also discuss common pitfalls and lessons learned. In response to participant feedback, we will include experiences for troubleshooting common problems

We surveyed the participants after the spring CFT bootcamp. Participants were overwhelmingly positive about the materials and activities and had improved confidence in their understanding of AI projects. One respondent said, "The bootcamp makes me better understand how users are using the AI packages installed on our systems. It also expands my knowledge of how to run and examine parallel and distributed jobs, hardware requirements like memory, GPU, IB, etc when supporting AI work. I also got a chance to sharpen my python programming skills." Our SDFT session will complete before EduHPC. We will include our experiences from the fall SDFT bootcamp in our talk. We plan to offer both sessions again in spring 2023.

Acknowledgement

This work is sponsored by NSF award #2118250

IV. A GPU Programming Lesson in the Pedagogical Style of the Carpentries

By: Alessio Sclocco, Hanno Spreeuw, Mateusz Kuzak

A. Introduction

Graphics Processing Units (GPUs) are not anymore confined to the domain of computer graphics but are nowadays used to accelerate a wide range of applications, from artificial intelligence to science and engineering. GPUs are such an integral part of the current landscape of high-performance computing (HPC) that seven out of ten systems in the June 2022 TOP500 [7] are powered by them; among these GPU-powered systems, there is also Frontier, the first exascale supercomputer.

Being able to use these devices is therefore becoming even more important for students and software engineers alike, in particular the ones working in research and academia. However, most GPU programming courses are aimed at learners with a computer science background and require notions in computer systems that are not always available to students and practitioners in other fields.

Within the Netherlands eScience Center, we have been teaching the principles of GPU programming, both internally and externally, since at least 2015 [8]. When teaching we noticed that most learners without a background in computer science would struggle to follow the more theoretical parts of our course. Moreover, these learners would also find it difficult to interact with a typical HPC system via the terminal to complete the assignments.

The Carpentries [9] is a global community with a goal of teaching programming and data science skills to researchers, and to do so the Carpentries, and the communities around it, develop lessons based on the pedagogical principles presented in [10]. The Carpentries courses are not taught in the classic classroom style, structured around lectures with learners passively listening to the instructor, but by writing code together with the instructor and doing exercises. This approach is called participatory live coding [11], and it has proven very effective in teaching programming to novices and researchers without a computer science background, and therefore we decided to develop a new GPU programming course based on it.

In this paper we introduce the GPU programming lesson developed by the Netherlands eScience Center according to the pedagogical principles of the Carpentries; this lesson is

currently available in the Carpentries Incubator [12] under an open-source license.

B. GPU Programming Carpentries Lesson

The development of this lesson began in early 2021, within the first edition of the Carpentries lesson development study group [13], and since then the lesson has been taught four times, both online and in-person. The lesson is meant to be taught in one full day, but can be naturally split into two half-days; we prefer this latter approach when teaching online to avoid screen fatigue.

As mentioned in Section IV-A, the lesson is almost entirely hands-on, with frontal teaching confined mostly to the introduction, while for the majority of the course the learners type along the code with the instructors. The programming languages used in the lesson are Python and CUDA; Python is used both as a way to achieve GPU acceleration without too much effort, and to manage the GPU when writing CUDA code.

Requirements to participate in the course are therefore maintained reasonably low: we only ask learners to be familiar with programming in general, and Python and Numpy in particular, and do not require them to have previous experience in HPC. Furthermore, learners are not required to have a programmable GPU on their laptops or have access to a GPU remotely because the whole course is taught by writing code in a Jupyter notebook via the browser. We therefore only require them to have a laptop with a reasonably modern browser installed. The lesson has been tested and taught using both Google Colab and SURF's JupyterHub for education.

There are three identifiable parts in the lesson, with this structure loosely based on Minto's pyramid principle [14], so that the complexity of the concepts increases during the course, but each part provides valuable content for some of the learners. The three steps are the following and are based on different ways to use GPUs when writing software: 1) use GPU accelerated libraries, 2) accelerate Python code with decorators, and 3) write CUDA kernels.

The first part of the lesson is aimed at Python programmers that make extensive use of libraries in their code. We show them how they can improve the performance of the presented code by replacing Numpy and SciPy calls with equivalent CuPy functions. This part helps introduce important concepts, such as the memory separation between GPU and host, without overloading learners with details while at the same time providing practical benefits to beginner programmers. After this part participants should be able to recognize the potential performance gains of using GPUs and be aware that some of the libraries they use may be replaced with GPU accelerated ones.

The second part of the lesson focuses on accelerating arbitrary Python code using Numba, therefore allowing learners to use GPUs even if their code does not make use of external libraries or if those libraries are not the bottleneck. When this lesson is taught as an online course, we combine parts one and two and teach them on the first day of the course, leaving part three for the second day. After this part participants should be able to recognize the sections of their Python code that could benefit from GPU acceleration and use Numba decorators to execute those on the GPU without major code refactoring.

The third and final part of the lesson is dedicated to CUDA programming. In this last part, learners are taught how to write their own code in a GPU-specific language to have more flexibility and performance, at the cost of increased code complexity. To manage the GPU and the data transfers between the GPU and the host, we use the interface to CUDA provided by CuPy. In our experience, this interface is easy to explain to Python programmers and simple enough to not distract course participants from the goal of writing their GPU code using CUDA. After this part participants should be able to write their own simple CUDA kernels and execute them using Python while at the same time being familiarized with the different CUDA memories and the concept of data sharing and synchronization.

C. Lessons Learned

Feedback from participants is fundamental to improve the lesson, and therefore we conclude every day of the course, be it a full in-person or a half online day, by asking each participant for feedback. Each learner is requested to provide one thing that he or she liked about the day and something that could be improved. It is based on this constant feedback from our students that in this section we present a short summary of the lessons learned while developing and teaching this lesson.

In a course like this, it is important to manage the expectations of the participants. At the Netherlands eScience Center, we provide this and other courses, for free, to a variety of students and researchers in the country, from different fields, of different ages, and from different backgrounds. We cannot make each of them a GPU expert in a few hours. Instead, we focus on giving them actionable skills and an understanding of how to keep growing from this point.

We decided to use CUDA, although proprietary and limited to NVIDIA GPUs because it is the most known and used GPU programming language. However, during each course, we are asked about solutions that are not limited to a single vendor. We try to keep this course general enough that the concepts, if not the details, that we teach can be easily translated to other languages such as OpenCL and HIP, and plan to develop extensions of the last part based on other languages.

Not requiring the participants to have GPUs available and instead basing the whole course on the access to Jupyter notebooks executed on an external cloud service has been highly praised by students. In this way, we avoid having to deal with heterogeneous environments and long setup and debugging time while at the same time providing a solution that is familiar also to the less experienced participants. At the end of the day, the learners can download a copy of the notebook they have been working on and run it on their own or on another system later.

One issue we have seen various learners struggle with is the CUDA syntax. Only a handful of the students and software

engineers that took part in our courses had experience in C, and even if they had, they may not have written C code in a while. As a result, they often struggle with the exercises that require them to write CUDA code. To alleviate this issue, if students are struggling with the syntax we provide, as an optional hint, a version of the same algorithm in standard C.

D. Conclusions

In this paper, we introduced the GPU programming lesson that the Netherlands eScience Center developed according to the Carpentries principles. The lesson is meant as an introduction to GPU programming for students, researchers, and software engineers from scientific fields other than computer science.

The lesson is based on a participatory live coding approach where learners write code together with instructors, and teaches participants how to 1) use GPU accelerated libraries, 2) accelerate Python code with decorators, and 3) write simple CUDA kernels.

The lesson is currently available, with an open-source license, in the Carpentries Incubator and has been taught online and in-person since 2021.

V. Peer-Review Assignments for a New Generation of Research Software Engineers

By: Pratik Nayak, Marcel Koch, Fritz Göbel, Hartwig Anzt

The progress of science and research in the age of computing has been accelerated by two things that have been co-designed and co-developed: Hardware, enabling massive computing power with GPUs, CPUs and scaling them to form massive supercomputers, and software, that forms the interface that processes scientific ideas into meaningful data to enable new discoveries.

In the recent past, hardware architectures have been specialized, and this specialization has led to fragmentation, which in turn has led to a myriad of programming paradigms, especially for parallel and high performance computing. To be able to run software on large high performance machines, one might need knowledge of GPU programming for various architectures, distributed programming with MPI and various other generic techniques such as task based programming.

While the knowledge and mastery of the above programming paradigms is important, in many cases without a focus on software quality, reproducibility and maintainability, the job of an RSE is incomplete. There are various tools that can improve productivity for an RSE. The use of a version control system in conjunction with well-defined continuous integration (CI) workflows can allow for easy tracking and bug detection in large software code bases. Additionally, adopting a workflow for collaborative software development using version control systems and web-based peer review platforms such as GitLab and GitHub can greatly improve software quality while encouraging contributions from various developers.

In our Numerical Linear Algebra for High Performance Computing (NLA4HPC) course at the Karlsruhe Institute of Technology [15], we aim to educate students in these essential RSE practices of version control, continuous integration, unit testing and peer review based workflow, in addition to teaching them parallel programming concepts and numerical linear algebra.

A. The Workflow

Our objective with the assignment was to integrate RSE practices into the assignments to demonstrate to the students, not only the ease of usage of these practices, but also the benefits of incorporating these in their own projects. The NLA4HPC course covers programming paradigms such as GPU programming with CUDA/HIP, multicore CPU programming with OpenMP and some basic distributed programming with MPI. We also cover numerical linear algebra concepts ranging from matrix-matrix multiplication, sparse matrix operations to direct and iterative solvers. The course is distributed over one semester over approximately 12 lectures and 12 exercise sessions. The exercise sessions can be used to discuss a reference solution with the students, or to give individual feedback.

The grade for the course involves the coding assignments, one end-term project and an oral exam.

B. The Setup

There are 6 coding assignments, covering the core concepts of the course, and are incremental in nature. As the focus of the course is to teach numerical linear algebra in the context of high performance computing, these assignments ask the students to implement building blocks which they can use in their own projects or other application codes they want to contribute to. Hence, the exercises generally consist of:

1) HW0: An introductory exercise to familiarize with the framework (not graded).
2) HW1: Single threaded dense BLAS operations: norm2, matrix-vector and matrix-matrix multiplication.
3) HW2: Parallel Dense matrix vector multiplication with OpenMP and CUDA.
4) HW3: Dense LU factorization using OpenMP tasks.
5) HW4: Compressed sparse row (CSR), SpMV with OpenMP and CUDA.
6) HW5: Conjugate gradient iterative solver with OpenMP and CUDA.

A constant feedback from the previous offerings of our course was that students were spending a significant amount of time on the build and compiling setup and bugs related to those. To allow the students to focus more on the kernel implementation and performance tuning rather than on the build system, we provided them with a complete framework, including scaffolding and a CMake setup.

Our next objective was to encourage students to write unit tests to ensure code correctness and easily figure out bugs in their code. Using the GoogleTest testing framework, we encouraged students to write unit tests for each kernel's edge case, providing basic unit tests which their kernels needed to pass to be valid. This testing framework was also built transparently within the build system.

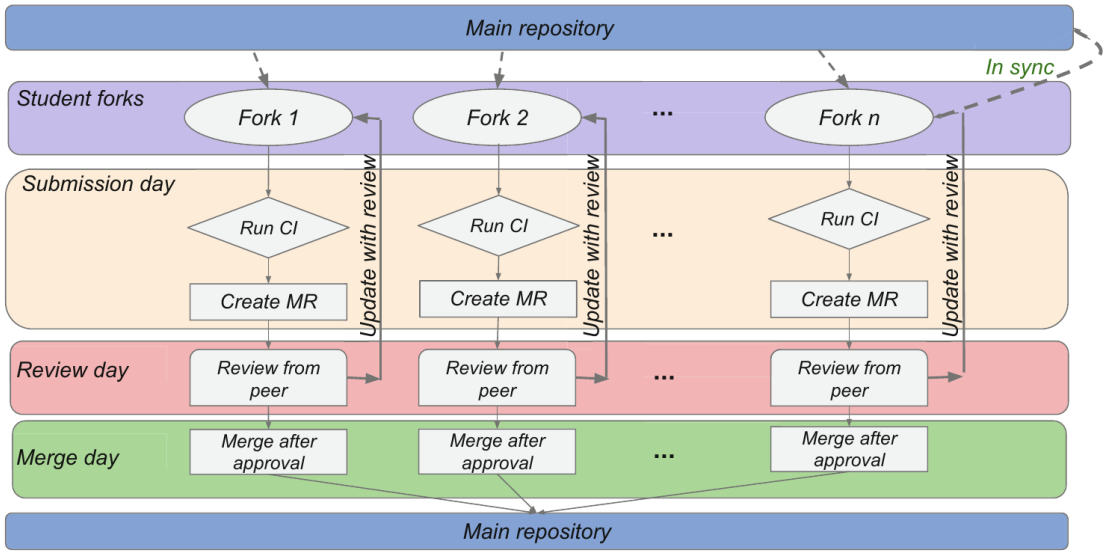

Fig. 1. The homework assignment workflow

Benchmarking was another important aspect that we wanted to familiarize the students with. Most exercises required them to measure the performance of their implementations and provide an analysis with a report. To ease their burden, within the framework, we added support for Google's benchmark and provided benchmarking capabilities. Because the students are usually not familiar with relevant problem sizes for each assignment, it is still necessary to provide them sensible ranges for it.

C. Bringing It All Together with Continuous Integration and Peer review

In addition to the multitude of tools and libraries we provided the students with, it was essential to automate the process of building the code and running their tests to emulate the workflow of a real research software library. To this end, we used the GitLab platform to provide CI support and allow students to just push to the repository and see if their code runs on the GitLab runners. In some cases, the students were not able to use the GitLab runners, and we provided custom runners on our machines to make sure that each student had CI access.

Finally, to bring it all together, we asked the students to create Pull Requests/Merge Requests (PR/MR) to the central repository from their forks. We assigned students to review each other's code, allowing them to look at their peers' code, give critical feedback, gain experience at code review and encouraged them to incorporate the feedback into their code. Figure 1 shows the workflow schematic.

D. Student Feedback

The main audience for the course were Masters students in Mathematics and Computer Science and some Bachelors students who have had some previous courses in linear algebra and programming in C/C++. We expected the students to have some familiarity with C/C++ programming and some basic knowledge of computing. To familiarize the students with the tools and libraries used in the course, the initial exercise sessions consisted of hands-on sessions on using git, writing unit tests with GoogleTest, writing benchmarks with Google's benchmark and getting familiar with the CI and peer review workflow on GitLab. I shows the feedback we received from the students. The overall feedback was positive, which has encouraged us to continue this practice in our future course offerings.

TABLE I
STUDENT FEEDBACK (1-STRONGLY AGREE TO 5-STRONGLY DISAGREE)

Statement	Avg Rating (1–5)
It was easy to use the framework	1.8
The exercises instructions and documentation was clear	1.5
It was easy to compile and run the code as provided	2.6
The code review from my peer was useful	1.6
The reviewing process was easy	3.3
I would like to see this type of framework adopted by other courses	1

E. Variants and Adapting to Other Courses

We believe that incorporating these practices into courses with coding assignments can encourage students to contribute to open-source software, adopt these practices in their own projects, preparing them for careers in sustainable programming. The core of the framework, including the CI, can be adapted to any other course. Most web-based platforms such as GitLab and GitHub provide computing resources for open source projects, and in case special hardware is needed, it is very easy to enable the CI to run on self-hosted machines.

REFERENCES

[1] "The European High Performance Computing Joint Undertaking." [Online]. Available: https://eurohpc-ju.europa.eu/

[2] "European Master For High Performance Computing." [Online]. Available: https://eumaster4hpc.uni.lu/

[3] C. Aasheim, J. Shropshire, L. Li, and C. Kadlec, "Knowledge and skill requirements for entry-level IT workers: A longitudinal study," *Journal of Information Systems Education*, vol. 23, no. 2, pp. 193–204, 2012.

[4] G. Zheng, C. Zhang, and L. Li, "Practicing and Evaluating Soft Skills in IT Capstone Projects," in *Proceedings of the 16th Annual Conference on Information Technology Education*, ser. SIGITE '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 109–113. [Online]. Available: https://doi.org/10.1145/2808006.2808041

[5] W. Groeneveld, B. A. Becker, and J. Vennekens, "Soft Skills: What Do Computing Program Syllabi Reveal About Non-Technical Expectations of Undergraduate Students?" ser. ITiCSE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 287–293. [Online]. Available: https://doi.org/10.1145/3341525.3387396

[6] N. Berente, J. Howison, J. Cutcher-Gershenfeld, J. L. King, S. R. Barley, and J. Towns, "Professionalization in cyberinfrastructure," *Available at SSRN 3138592*, 2017.

[7] "Top500 june 2022." [Online]. Available: https://top500.org/lists/top500/2022/06/

[8] B. van Werkhoven, "Netherlands escience center gpu programming course." [Online]. Available: https://github.com/benvanwerkhoven/gpu-course

[9] "The carpentries." [Online]. Available: https://carpentries.org/

[10] S. A. Ambrose, M. W. Bridges, M. DiPietro, M. C. Lovett, and M. K. Norman, *How Learning Works: Seven Research-Based Principles for Smart Teaching*. John & Sons: Wiley, 2010.

[11] A. Nederbragt, R. M. Harris, A. P. Hill, and G. Wilson, "Ten quick tips for teaching with participatory live coding," *PLoS Comput Biol*, vol. 16, p. 9, 2020.

[12] "Carpentries incubator gpu programming course." [Online]. Available: https://github.com/carpentries-incubator/lesson-gpu-programming

[13] "The carpentries lesson development study groups." [Online]. Available: https://carpentries.org/blog/2020/12/lesson-development-study-groups/

[14] B. Minto. The pyramid principle, 1981.

[15] P. Nayak, F. Göbel, and H. Anzt, "A Collaborative Peer Review Process for Grading Coding Assignments," in *Computational Science – ICCS 2021: 21st International Conference, Krakow, Poland, June 16–18, 2021, Proceedings, Part VI*. Berlin, Heidelberg: Springer-Verlag, Jun. 2021, pp. 654–660. [Online]. Available: https://doi.org/10.1007/978-3-030-77980-1_49