Designing a MOOC for EUMaster4HPC: Emphasizing OpenACC, OpenMP Offloading, and the HIP Programming Models

Ezhilmathi Krishnasamy* and Pascal Bouvry

Department of Computer Science, FSTM, University of Luxembourg, Belval, Luxembourg
Email: ezhilmathi.krishnasamy@uni.lu (E.K.); pascal.bouvry@uni.lu (P.B.)
*Corresponding author
Manuscript received February 17, 2025; revised March 3, 2025; accepted April 18, 2025; published July 8, 2025

***Abstract*—In the rapidly evolving landscape of education, the integration of digital technologies has led to the emergence of Massive Open Online Courses (MOOCs), which are becoming increasingly available across various educational domains. Despite a prevailing preference for traditional educational methods, MOOCs present viable alternatives that complement rather than replace conventional classroom instruction. The EUMaster4HPC is a newly established pan-European master's program that focuses on High Performance Computing (HPC). This consortium aims to develop a distinctive curriculum centred around modular education in HPC. Additionally, EUMaster4HPC is committed to promoting its own MOOC offerings. In this paper, we provide a detailed analysis of design strategies for producing MOOC focused on OpenACC, OpenMP Offloading, and Heterogeneous-Compute Interface for Portability (HIP) programming models, all of which are crucial for accelerating computations on GPUs. These models emphasize high-level parallel programming, while HIP also addresses low-level programming, necessitating familiarity with CUDA for effective learning. Furthermore, the discussed programming models support heterogeneous programming, allowing for the simultaneous utilization of both CPU and GPU resources. This work investigates the instructional design methodology, course content, structure, and strategies employed in this MOOC, thereby contributing to the broader discourse on digital education in HPC.**

***Keywords*—Massive Open Online Course (MOOC), instructional design, computer science, parallel programming, Open Accelerators (OpenACC), Open Multi-Processing (OpenMP) offloading, Heterogeneous-Compute Interface for Portability (HIP), Compute Unified Device Architecture (CUDA)**

I. Introduction

The EUMaster4HPC [1] is a pan-European master's program established with funding from the EuroHPC initiative. This program offers up to four specializations: System Architecture, System Development and Support, Performance Analyst and Advisor, and Numerical and Data Analysis for Scientific Domains [2]. The consortium provides guidelines for the curricula of both the first and second years of study. In total, eight universities offer master's degrees in HPC. Students are required to study at two different universities, with the first year spent at one institution and the second year at another. This unique curriculum structure is complemented by a variety of learning activities. Notably, the program includes two technical challenges that span an entire semester, a two-week summer school, internships, and opportunities for students to participate in HPC workshops, such as the EuroHPC Summit and ETP4HPC.

Table 1. Comparison of GPU programming courses

Features	**Udacity** [3]	**Coursera** [4]	**EUMaster4HPC**
GPU Architecture	X	X	X
CUDA	X	X	X
OpenACC	-	-	X
OpenMP Offloading	-	-	X
HIP	-	-	X
Performance Analysis	X	-	X
Software Integration	-	X	-
Using Libraries	-	X	-
Learning Through	Image Processing	Image Processing	BLAS Routines

The following are key aspects of the EUMaster4HPC program:

- Establishing a modular curriculum that enhances the visibility of a Master of Science in HPC, thereby increasing its attractiveness to prospective students.
- Promoting the establishment of new educational programs and improving existing ones through strong European collaboration.
- Involving the private sector in realizing the modular curriculum and providing training opportunities for students.

In addition to the traditional classroom curriculum, the EUMaster4HPC consortium actively promotes Massive Open Online Courses (MOOCs) for students pursuing a master's degree in HPC[1]. The consortium has categorized the courses into three main sets: fundamentals, balancing, and specialization models. These MOOCs, developed specifically under the EUMaster4HPC initiative, fall into one of those categories.

It is essential to note that although these MOOCs are designed as free-credit courses, students will not receive European Credit Transfer and Accumulation System (ECTS) credits upon completion. However, the awarding of ECTS may vary by university, where certain institutions may encourage students to enrol in specific courses and grant ECTS credits contingent upon successful course completion. For instance, students may be required to undergo an examination conducted by faculty from their respective universities, which could include both oral and written components based on the MOOC content.

Nevertheless, it is crucial to clarify that the MOOCs themselves do not provide standardized examinations or evaluation metrics for students. This approach allows

[1] http://1https//moodle.eumaster4hpc.eu/

flexibility for universities, enabling them to adopt the courses in ways that align with their specific evaluation criteria and teaching methodologies. Thus, the EUMaster4HPC initiative aims to offer universities the freedom to tailor their student evaluation processes while utilizing the developed courses.

Although various MOOCs from awarding universities provide content on a wide range of topics, this paper focuses specifically on MOOC designed for OpenACC [5], OpenMP Offloading [6], and Heterogeneous-Compute Interface for Portability (HIP) [7]. It also covers introductory material on parallel programming and methodologies for performance analysis.

A. Relevant Study

Numerous MOOC platforms, such as Coursera [8], edX [9], Udacity [10], and FutureLearn [11], offer a wide range of courses across various subjects, including parallel programming. For instance, Udacity provides courses in parallel programming, which include topics on CUDA [3]. The curriculum of these courses presents a diverse range of topics that differ significantly from the content of our MOOC course. Similarly, Coursera offers a series of courses in GPU programming, focusing specifically on the CUDA programming model [4, 12]. This curriculum comprises up to four courses, each with an estimated workload of 21 hours per course. The individual courses emphasize the utilization of both single and multiple Graphics Processing Units (GPUs) and incorporate libraries such as cuTensor and cuDNN. Furthermore, we have not identified any relevant or well-documented MOOCs that cover OpenACC and OpenMP Offloading. Therefore, we assert that our MOOC content differs from those listed in Table 1.

In addition to various online platforms, several universities worldwide also offer MOOCs, including the Massachusetts Institute of Technology (MIT) [13], Harvard University [14], Stanford University [15], and the University of London [16]. However, whether the courses adhere to the EUMaster4HPC guidelines necessary for the consortium to recommend them to students remains uncertain. This uncertainty was a primary motivation for EUMaster4HPC to establish its own MOOC platform, which would align with the guidelines developed by the consortium and implemented in the EUMaster4HPC MOOCs. Moreover, EUMaster4HPC MOOCs are offered free of charge to students, unlike other platforms that charge tuition fees.

Apart from the course contents, various platforms also offer hosting for MOOCs, including Open edX [17], Moodle [18], Canvas [19], and Google Classroom [20]. Some of these hosting platforms are proprietary, while others continue to provide open-source software options (see https://oli.cmu.edu/). However, within the context of the EUMaster4HPC initiative, we utilize Moodle for hosting our MOOC materials.

B. Our Contributions

This section provides a brief overview of our achievements in developing MOOC content on OpenACC, OpenMP Offloading, and HIP.

- Selection Criteria: We elucidate the criteria used for selecting course content and learning outcomes to meet the MOOC requirements established by EUMaster4HPC.
- Organization of Learning Materials: We detail the structuring of learning materials within the MOOC framework, which includes videos, assessment questions, and scholarly articles designed to clarify the concepts inherent to the MOOC content.
- Quality Standards Guidelines: We provide comprehensive guidelines for producing MOOC content, ensuring a seamless learning experience with the GPU programming model for those pursuing a career path in scientific computing.

II. Course Structure of EUMaster4HPC

The EUMaster4HPC is a pan-European master's program involving a consortium of diverse partners, including universities, HPC research centres, public institutions, and private enterprises.

A. Curriculum Design

The consortium has meticulously developed a curriculum that addresses the learning outcomes for both the first and second years of the program, comprising two semesters each. This process involved an extensive analysis of current academic and industrial needs, spanning several months and including multiple iterations. Each topic is carefully selected to focus on specific areas, such as the parallel programming module offered in the first-year coursework. Fig. 1 illustrates the EUMaster4HPC programme outline of the modules in the entire programme, while Figs. 2 and 3 display the list of modules that were proposed within the EUMaster4HPC curriculum.

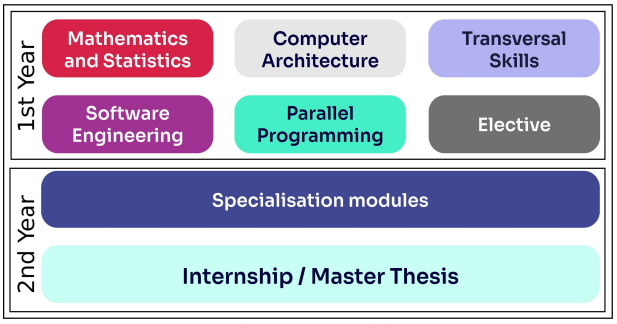

Fig. 1. Basic curriculum structure of the EUMaster4HPC (adopted from [1]).

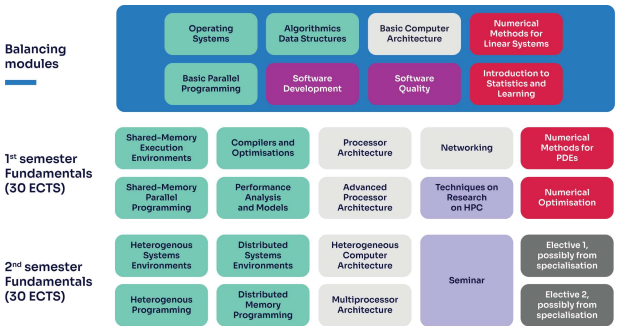

Fig. 2. First-year curriculum for EUMaster4HPC (adopted from [1]).

The EUMaster4HPC curriculum is structured around five main categories of educational blocks:

- **Balancing Model:** This block provides foundational courses in mathematics, parallel programming, software engineering, and computer architecture.
- **Semester One:** This semester establishes a basic understanding of core concepts and principles.

- **Semester Two:** This semester builds on the fundamentals, introducing slightly more advanced topics.
- **Semester Three:** This semester is dedicated to specialized topics tailored to students' interests and career goals.
- **Semester Four:** This semester focuses on the master's thesis, allowing students to conduct in-depth research in their chosen area of study.

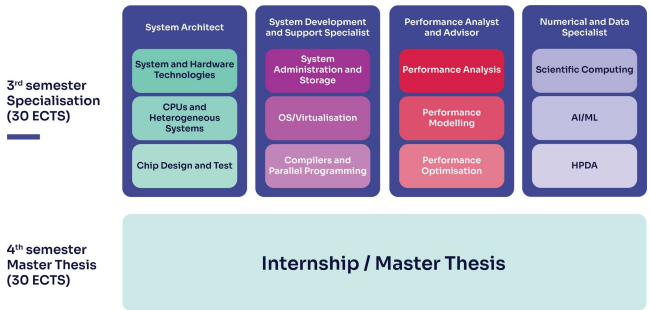

Fig. 3. Second-year curriculum for EUMaster4HPC (adopted from [1]).

B. Why Promote MOOCs within EUMaster4HPC

Traditionally, HPC has been viewed primarily as a discipline within computer science, attracting students with backgrounds in this field. However, this trend has shifted over the past two decades. Today, HPC is increasingly appealing to individuals with various STEM educational backgrounds, allowing those with degrees in mathematics, physics, and engineering to pursue a master's degree in HPC. One of the primary objectives of the EUMaster4HPC program is to encourage students from diverse backgrounds, particularly those in mathematics, physics, and engineering, to pursue studies in HPC.

Despite this encouragement, these students often face challenges, particularly when learning subjects such as parallel programming, computer architecture, and software engineering, which may not have been central to their prior education. To tackle this issue, EUMaster4HPC has proposed a "balancing model," illustrated in Fig. 2. This model allows students to enroll in specific courses during their first year to acquire necessary skills that may be lacking. By implementing this approach, the model aims to minimize knowledge gaps and facilitate a smoother transition for students, helping them successfully complete their master's program.

The balancing model can be studied through courses offered at existing universities and via the EUMaster4HPC MOOCs. For instance, a bachelor's graduate in mathematics from University A might need foundational knowledge in parallel programming before applying for master's studies at University B. Therefore, a student can attend the parallel programming courses offered by University B or engage with the EUMaster4HPC MOOCs at their own pace. This flexibility is one of the key advantages of MOOCs; they offer students a choice between traditional classroom instruction and online learning, providing opportunities to learn at their own pace. Moreover, the EUMaster4HPC MOOC is not only an optional component of the balancing model but can also serve as an additional topic covered in the first year (semesters 1 and 2) of the EUMaster4HPC program.

III. Course Design Strategies

MOOCs were initially introduced in 2008, and by around 2012, more universities and MOOC providers, such as Udacity and edX, became involved in offering these courses [21]. MOOCs can be categorized into various types based on methodology and learning approaches [22–24]. However, we highlight three important categories as follows:

- Connectivist MOOCs (cMOOCs) are centered around the principles of connectivity, networking, and collaborative knowledge creation. This model promotes peer-to-peer interaction, enabling participants to engage actively in the learning process. An early example of this approach is the Connectivism and Connective Knowledge (CCK08) course.
- Task-Based MOOCs (tMOOCs) focus on the completion of specific tasks as a means of engagement. In these courses, assessments are typically conducted through projects and practical assignments. Platforms that emphasize project-based learning usually offer this type of MOOC, providing learners with hands-on experience.
- Extended/Traditional MOOCs (xMOOCs), commonly referred to as xMOOCs, draw upon cognitive learning theories and present course content through various formats such as videos, slides, articles, and quizzes. The assessment methods for these courses predominantly involve examinations and quizzes. Prominent examples of xMOOC providers include Udacity and edX, and the MOOC discussed in this paper is based on the xMOOC model.

In addition, modes of E-learning or online learning can be categorized into three types, as follows [25]:

- Synchronous Learning: In this mode, participants interact with instructors in real-time during live sessions or face- to-face learning sessions.
- Asynchronous Learning: In this approach, participants do not need to be online simultaneously with the instructors. They can learn at their own pace through recorded videos and teaching materials that are accessible anytime. Our MOOC is based on this approach.
- Hybrid Learning: This mode combines both synchronous and asynchronous learning.

A. Instructional Design Adoption

There has been a considerable amount of instructional development in MOOC, and academic course design has been developed and studied [26–28]. For our MOOC, we selected the ADDIE model [29] and Merrill's First Principles [30], which is also depicted in Fig. 4. These frameworks are widely used for the development of MOOCs and other academic educational courses. We utilized the ADDIE model for course design while incorporating Merrill's First Principles to enhance the working examples within the course.

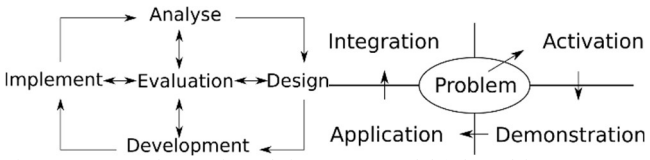

Fig. 4. Framework overview: (left) ADDIE model (adopted from [29, 31]); (right) Merrill's first principles of instruction (adopted from [29, 31]).

The ADDIE model provides a comprehensive framework comprising five stages for designing academic courses and MOOCs. While each methodology encompasses a range of procedures, including project management and logistical considerations, we will not discuss generic topics such as project management and logistics in this context. Therefore, the following discussion highlights only the essential technical aspects of producing this MOOC.

- **Analysis**: This approach identifies the knowledge gaps based on current scenarios in academia, industry, and research. Previous analyses have incorporated various interviews with experts from industry, academia, and research institutes, as well as job market assessments (e.g., job profiles and required skill sets) and emerging technology trends. Based on this comprehensive analysis, EUMaster4HPC has proposed a curriculum, which is illustrated in Fig. 2. The curriculum emphasizes course modules on parallel programming, including shared memory systems, distributed systems, and heterogeneous systems.
- **Design**: During this stage, we have constructed learning outcomes and course delivery options. Section VI provides a detailed explanation of the course delivery approach we have adopted for this MOOC, such as videos, articles, slides, and quizzes. Additionally, we drew insights from our experiences in classroom teaching, online teaching [31], and previous MOOC [32] on parallel programming, specifically focusing on CPU and GPU parallel programming. These insights provided constructive ideas for the content of this MOOC. In the context of MOOC delivery, selecting an appropriate hosting platform, commonly referred to as a Learning Management System (LMS), is crucial for effective delivery. Given the array of MOOC hosting platforms available [33], we opted for Moodle [34]. This decision was primarily influenced by Moodle's user-friendly graphical interface, which is comparable to that of other proprietary platforms, as well as its ease of maintenance. Additionally, it offers good flexibility, allowing for an increasing number of students over time, ensuring that even if many students attempt to access Moodle simultaneously, they will still experience a positive user experience.
- **Development**: This section is dedicated to the tools utilized for creating the MOOC material. For this course, we employed a standard video recording option and provided subtitles alongside the videos. Additionally, Moodle supports various formats, including PDF, and offers an interface for hosting quizzes with multiple-choice options. Finally, articles can be integrated using Markdown text [35]
- **Implementation**: This refers to the delivery of our MOOC. Learners are notified of the course's availability on Moodle prior to its commencement. Since this MOOC is based on an asynchronous model, the EUMaster4HPC students can engage with the course material at their own pace. During this stage, we analyze the course content both in the pre-production phase and throughout the course run, gathering input from experts and learners. Although various tools are available for analyzing MOOC content [36], such as OLC Dimensions of Quality [37] and Quality Matters [38], we do not rely solely on their support. Instead, we have adapted their methodologies where necessary to prepare questionnaires for learners and experts, allowing us to collect feedback on the course. Section VII provides the feedback received from students upon course completion.
- **Evaluation**: During this phase, we implemented necessary revisions (e.g., adjusting video quality) based on the feedback received during the pre-production phase. Course materials, including videos, slides, articles, and quizzes, were shared with experts in parallel programming who possess significant pedagogical experience. Additionally, the MOOC has a working group within EUMaster4HPC that oversees the quality of the MOOC content, ensuring it meets both quality standards (for videos, slides, and article text) and the high-level learning objectives. Furthermore, feedback from participants will be analyzed subsequently, and critical recommendations will be incorporated into future iterations of the course. We plan to update the course every semester, with the next update scheduled before the autumn 2025 semester, if necessary.

Although we used the ADDI model to formulate the entire course structure, we aimed to incorporate Merrill's First Principle to guide the development of the course content. Below is an outline of how we have adopted Merrill's First Principle for our MOOC:

- *Problem-centered*: Introduce concepts of key Application Programming Interfaces (APIs) in the GPU programming model, as this will help in solving problems later on.
- *Activation*: Gradually present the Basic Linear Algebra Subprograms (BLAS) routines as challenges to be solved in parallel using the GPU programming model. And how the GPU programming model is effectively used to parallelise the BLAS routines on the GPU.
- *Demonstration*: Start by showing how to construct GPU kernel calls and gradually instruct how to transfer data from the CPU to the GPU. Finally, provide guidance on how to parallelize the GPU kernels using the APIs of the respective GPU programming model.
- *Application*: Encourage students to solve various BLAS routines based on what they have learned.
- *Integration*: Motivate students to explore how GPU models can be applied beyond BLAS routines. For example, we will pose a set of open questions, such as solving stencil code (wave propagation), particle-in-cell code (plasma physics), 1D or 2D finite element methods (computational solid mechanics), and 1D or 2D finite volume methods (computational fluid dynamics).

More importantly, EUMaster4HPC proposes high-level learning outcomes from MOOCs. It is therefore up to the course providers to determine the type of content to include in the MOOC. The course providers, who are experts in this field and have been actively conducting research for the past decades, are well-equipped to create a comprehensive course on GPU programming models.

IV. Analysis of MOOC Content

The EUMaster4HPC program is distinctive in its

comprehensive curriculum, focusing on parallel programming. To enhance our educational offerings, we have developed a MOOC that encompasses OpenACC, OpenMP Offloading, and HIP. These topics are integral to the foundational knowledge and learning outcomes of the EUMaster4HPC, and the decision to include them was informed by their relative underexposure in comparison to CUDA, which has been extensively covered in recent years.

A. Motivation

The importance of learning OpenACC, OpenMP Offloading, and HIP stems from the substantial role that GPUs and other accelerators play in modern supercomputing environments. According to the TOP500 [39] list, many supercomputers globally are equipped with diverse accelerators, including those from AMD, Nvidia, and Intel. Table 2 shows key differences between these GPUs and how these architectures are designed. This diversity underscores the necessity of mastering various GPU programming models to effectively utilize these technologies. While numerous programming models exist, such as low-level programming (OpenCL [40] and CUDA) and high-level programming, as well as templated library options, this study posits that a focused learning approach on both low-level and high-level programming models is sufficient. The objective should be to gain proficiency in these models to enable targeting of a variety of GPU architectures. For instance, the concepts covered in our MOOC course on OpenACC, OpenMP Offloading, and HIP can facilitate the parallelization of any scientific applications across AMD, Nvidia, and Intel GPUs.

Table 2. Comparison of various GPU vendors.

Feature	Intel	NVIDIA	AMD
GPU	Xe-HPC Max 1550 [41]	A100 [42]	Instinct MI250X [43]
Execution Units	512 × 2 XVEs	108 SMs	220 CUs
Threads per Unit	8	64 CUDA cores per SM	64 ALUs per CU
Total Threads	4096 × 2	6912	14080

Note: CUs = Compute Units; SM = Streaming Multiprocessors; XVEs = Xe Vector Engines; ALU = Arithmetic Logic Unit.

Table 3. Comparison of important GPU programming models.

API	Programming Style	Ease of Use	Performance	Primary Vendor Support
CUDA [12]	Language (C/C++ and FORTRAN) extensions	Medium	High	NVIDIA
HIP [7]	Language extensions (C/C++ and FORTRAN)	Medium	High	AMD and NVIDIA
OpenCL [40]	Language extensions (C/C++)	Low/Medium	Medium	Cross-vendor
OpenACC [5]	Directive Based	Medium/High	High/Medium	NVIDIA and AMD
OpenMP Off. [6]	Directive Based	Medium/High	High/Medium	Intel, AMD, and NVIDIA
SYCL [44]	Templated based	Low/Medium	Medium	Intel, AMD, and NVIDIA
Kokkos [44]	Templated based	Low/Medium	Medium	Intel, AMD, and NVIDIA
RAJA [45, 46]	Templated based	Low/Medium	Medium	Intel, AMD, and NVIDIA
Thrust [47]	Templated based	Low/Medium	Medium	NVIDIA

Although performance discrepancies may arise between different architectures, these variations are typically manageable. A pertinent question is why templated library programming, such as Kokkos [48] or RAJA [49], has not been emphasized. Templated library programming, which is generally categorized as high-level programming, often limits programmers' control over their applications. This limitation can hinder a comprehensive understanding of application performance and prevent the full utilization of GPU hardware. Therefore, acquiring a solid foundation in both low-level and high-level programming models will position learners to more effectively engage with templated programming approaches in the future. This sequential understanding enhances one's ability to optimize applications while leveraging the capabilities of diverse GPU architectures. Furthermore, Table 3 presents a systematic overview of the widely recognized GPU programming models and their feasibility across various GPU architectures. This clearly highlights that learning HIP, CUDA, OpenACC, and OpenMP Offloading would help achieve offloading to scientific computations on any GPUs.

B. Course Contents

The MOOC content is structured into five comprehensive sections, each aligning with the recommended learning outcomes for the EUMaster4HPC program:

- Introduction to General-Purpose GPU (GPGPU) Programming: This section provides a foundational understanding of GPGPU programming. It covers the motivation for learning parallel programming, essential concepts in parallel processing, and an overview of CPU and GPU architectures. Additionally, it details the organization and structure of the course, ensuring participants grasp the course framework from the outset.
- OpenACC: This section begins with a thorough introduction to the essential APIs associated with OpenACC. It progresses from basic concepts to intermediate-level topics, equipping learners with the knowledge needed to effectively utilize OpenACC for parallel programming. Key learning outcomes include the application of OpenACC directives and the ability to optimize code for accelerated execution.
- OpenMP Offloading: This section initially provides a brief overview of the OpenMP programming model. It then delves into fundamental concepts, emphasizing the critical aspects of data management. The curriculum advances to intermediate-level topics on OpenMP offloading, enabling learners to understand how to create computational blocks and effectively implement strategies for offloading computations from CPU to GPU.
- HIP: This section introduces the HIP programming model, starting from basic concepts to more advanced topics. A significant focus is placed on the transition from CUDA to HIP, with an in-depth discussion of the nuances involved in converting CUDA code into its HIP equivalent. This equips learners with the skills to adapt existing CUDA applications for broader compatibility and enhanced performance across heterogeneous platforms.

- Performance Analysis: This section covers the technical aspects and methodology for profiling and optimizing code effectively. While learning parallel programming and parallelizing code can be relatively straightforward, mastering optimization techniques for a specific application requires a good understanding of profiling tools. Therefore, this section provides an overview of various tools and options available for profiling code in OpenACC, OpenMP Offloading, CUDA, and HIP.

This structured approach encourages a systematic progression through complex topics, emphasizing the relevance of each subject to the overarching goals of high-performance computing education. The following sections offer additional brief explanations of the course content introduced in this MOOC.

C. GPU Programming in OpenACC

OpenACC is a programming model designed for use with C/C++ and Fortran languages, allowing for both GPU and CPU computing, as well as a hybrid combination of the two. It is recognized as a high-level parallel programming model, contrasting with lower-level paradigms such as CUDA and OpenCL. A significant advantage of OpenACC is its accessibility; programmers familiar with C/C++ and Fortran can seamlessly port their existing code to utilize accelerators. Furthermore, OpenACC currently supports both Nvidia and AMD GPUs. However, it is noteworthy that, due to the evolving nature of the technology, only the Fortran implementation of OpenACC currently supports AMD GPUs, leaving C/C++ users without this capability at present.

D. GPU Programming in OpenMP Offloading

OpenMP Offloading relies on a directive-based programming model, which supports both C/C++ and FORTRAN languages. Traditionally, OpenMP has focused on facilitating shared memory programming. However, the introduction of the OpenMP 4.0 specification marked a significant advancement by incorporating offloading capabilities that enable computations to be executed on GPUs. This functionality is compatible with Nvidia, Intel, and AMD GPUs, thereby broadening its applicability within high-performance computing environments. Importantly, in contrast to OpenACC, OpenMP Offloading provides comprehensive support for both C/C++ and FORTRAN across Nvidia and AMD hardware. This feature positions OpenMP Offloading as a versatile option for developers seeking to leverage GPU acceleration in their applications.

E. GPU Programming in HIP

HIP represents a low-level programming model designed to support both C and C++ programming languages. A notable advantage of HIP is its capability to execute on both Nvidia and AMD GPUs concurrently. By functioning as a low-level programming model, HIP is expected to deliver superior performance across various hardware platforms, thereby serving as a valuable tool for developers seeking to enhance portability and efficiency in GPU computing.

F. Performance Analysis

In the early stages of learning parallel programming or writing parallel code, developers may mistakenly believe their code is well-optimized or free of issues such as memory leaks. To eliminate these uncertainties, it is essential to familiarize oneself with the tools and techniques available for profiling code on GPUs. Typically, these profiling tools offer both Graphical User Interface (GUI) and command-line options, allowing users to collect various events, metrics, and trace function calls. This data provides crucial insights into several aspects of application performance, including memory usage, CPU and GPU occupancy, memory transfer between the CPU and GPU, compute kernel efficiency, and API call utilization. By leveraging these profiling tools, developers can gain a deeper understanding of their code's performance characteristics and identify potential areas for optimization.

V. Selection of Core Course Contents

To design the course effectively, it is essential to first identify relevant topics and subsequently define corresponding learning outcomes that align with the guidelines established by EUMaster4HPC. The following section presents a comprehensive description of how the content for each selected topic is integrated within the course framework.

A. OpenACC

OpenACC, as a directive programming model, requires a comprehensive understanding of its fundamental APIs to effectively parallelize code. This framework offers several critical APIs that cater to key areas of interest, which are outlined below:

- Parallel Constructs:

Kernel: This compute construct automatically parallelizes loops while prioritizing safety, in cases where loop dependencies exist or complexities arise. Therefore, it is advisable to utilize the kernel construct when managing intricate loop variables or when there is a limited understanding of parallelization.

Parallel: In contrast, the parallel construct provides customized options to meet specific parallelization requirements.

Loop Construct: The loop construct facilitates the parallelization of loops and incorporates several clauses designed to enhance execution efficiency, including *gang*, *workers*, *vector*, *collapse*, and *reduction*. These clauses are integral to the effective parallelization of computations, particularly the *collapse* clause, which is critical for optimizing nested loops encountered in matrix operations and various numerical algorithms. Furthermore, the *workers*, *vector*, and *collapse* clauses contribute to achieving a high level of parallelization that closely resembles the performance typically associated with low-level CUDA programming. Consequently, selecting a high-level programming paradigm still enables performance levels comparable to those attained through CUDA or other low-level programming approaches.

- Data Management: The primary goal of GPU computing is to offload computational tasks from the CPU to the GPU. This transition necessitates the transfer of data from the CPU to the GPU, as computations performed on the GPU depend on the availability of this data. OpenACC provides various data movement APIs crucial for efficient programming, including structured,

unstructured, and low-level options.

- Computational Linear Algebra: This chapter discusses how previously learned programming methodologies can be applied to solve fundamental linear algebra routines, such as dot products, vector addition, and matrix operations. These operations, while distinct from one another, are essential and widely utilized in computational science.

B. OpenMP Offloading

OpenMP offloading has garnered substantial interest within the scientific community following the introduction of computation offloading in the OpenMP specification 4.0. Since then, the OpenMP specification has evolved to incorporate more sophisticated and versatile APIs designed to facilitate efficient programming for accelerators. These advancements aim to achieve performance levels comparable to those of established frameworks, such as CUDA and OpenACC. The following items delineate the key topics addressed in the OpenMP Offloading chapter.

- OpenMP Overview: To adequately understand OpenMP Offloading, it is necessary to first introduce the OpenMP programming model. The OpenMP Offloading API includes extensions and modifications that distinguish it from standard OpenMP, necessitating a foundational grasp of the core OpenMP constructs and principles.
- Compute Constructs: OpenMP Offloading offers a diverse array of compute constructs that provide greater flexibility than those in OpenACC. Although the Nvidia HPC SDK is highly optimized for OpenACC, OpenMP Offloading is supported by several open-source compilers, such as GNU, which may require additional manual tuning of compute constructs to optimize performance during the execution of computational blocks. For instance, the directive `#pragma omp target teams` establishes a league of teams on the device executed by initial threads, while `#pragma omp target teams distribute parallel for` facilitates the offloading of code, distributes iterations across the teams, and enables parallel execution.
- Data Mapping: A thorough understanding of data mapping is pivotal in GPU programming, particularly when offloading computations to accelerators. Like OpenACC, OpenMP Offloading provides a comprehensive set of concepts for managing data transfers between the CPU and GPU. Therefore, it is imperative to master the fundamental aspects of data transfer to the GPU, as this knowledge significantly enhances performance outcomes in OpenMP Offloading.
- Computational Blocks: This section elaborates on the implementation of basic linear algebra routines through the application of the previously acquired concepts in OpenMP Offloading. Furthermore, it emphasizes the effective utilization of compute constructs and data mapping techniques to solve fundamental linear algebra problems without sacrificing performance efficiency.

C. HIP

HIP operates on a concept similar to that of APIs used in CUDA, meaning that mastering one is sufficient to understand both programming models. One of the advantages of HIP is its ability to execute on both Nvidia and AMD GPUs. Consequently, the content will focus on CUDA, as CUDA code can be converted into HIP using HIP compilers. The following items outline the essential topics covered in the HIP programming model in this MOOC.

- GPU Architecture and Essential APIs: It is crucial to understand the GPU architecture and how GPU cores are organized in both Nvidia and AMD GPUs. For instance, on Nvidia, it is essential to know how GPU cores are structured in the streaming multiprocessor and how memory is laid out, including the L2 cache, shared memory, and registers [50]. Similarly, AMD has its own organization for the cores, such as compute units with SIMD architecture [51].
- Threads Organization: A key aspect of low-level programming in CUDA and HIP is the control over threads and computational blocks. This concept is related to Single Instruction Multiple Threads (SIMT). Therefore, it is essential to have a solid understanding of thread organization when creating and invoking them in computational blocks, which may be structured in 1D, 2D, or 3D formats.
- CUDA to HIP: Understanding the methods for converting CUDA code to HIP is vital. Introducing this concept early in the course minimizes uncertainty when learning both CUDA and HIP, allowing students to focus on the core concepts. It covers the key concepts and techniques involved in converting CUDA code to its HIP equivalent.
- Memory Management: As with any GPU programming model, it is essential to know how to transfer data from the CPU to the GPU. This will be introduced through working examples, such as vector operations. While CUDA offers advanced memory transfer concepts, such as asynchronous data transfer using CUDA streams and peer-to-peer (P2P) memory copy, the course will focus on foundational concepts suitable for intermediate learners.
- CUDA Kernels: Computational GPU kernels are crucial for achieving performance in GPU programming; thus, understanding kernel implementation is essential. For example, it is necessary to learn how to properly align threads within a computational block.
- Unified Memory: CUDA offers a unified memory option that simplifies memory allocation for those who prefer not to manually manage data transfers. Although this approach may be the easiest way to program, it is essential to note that it can only be used with a single GPU.

D. Techniques and Tools for Performance Analysis

Since the course focuses on OpenACC, OpenMP offloading, and HIP, it is essential to concentrate on the tools and techniques used to profile these parallel programming methods. The following sections outline key topics covered in this context.

- OpenACC: It benefits from comprehensive compiler support through NVHPC [52], enabling the profiling of code on both CPUs and GPUs. NVHPC provides both GUI and command-line profiling options. While it is

beneficial to learn both methods, command-line profiling is typically more lightweight. Given that OpenACC offers numerous APIs, it is crucial to understand their efficient utilization for compute kernels, data transfers between CPU and GPU, and memory allocation.

- OpenMP Offloading: Although many open-source compilers, such as Cray [53] and GCC [54], are available, they are not as mature as NVHPC compilers. However, NVHPC is limited to Nvidia GPUs. Therefore, understanding NVHPC for Nvidia and Cray & ROCm for AMD GPUs provides a comprehensive framework for optimizing GPU codes across various architectures.
- HIP and CUDA: Nsight Systems [55] and Nsight Compute [56] offer comprehensive performance analysis tools for CUDA code. Similarly, for HIP code within the ROCm ecosystem [57], profiling options are available. Each profiling tool from NVIDIA and ROCm employs unique methodologies for evaluating code performance. Mastering these techniques will pave the way for optimizing low-level GPU programming. For instance, there are various strategies for optimizing compute kernels in CUDA and HIP, including efficient thread organization, memory utilization, and synchronization. Acquiring this knowledge will be highly beneficial for further code optimization.

The examples presented in this course are grounded in the fundamental subtopics of basic linear algebra. Scientific and engineering challenges are often addressed through differential equations, which ultimately lead to solving systems of equations. These systems can be solved either iteratively or directly, employing various BLAS routines. For instance, the conjugate gradient method relies on matrix-vector multiplication and vector dot products. Similarly, in artificial intelligence, matrix multiplication plays a crucial role. Consequently, learning parallel or GPU programming through BLAS routines proves to be both beneficial and applicable. Each BLAS routine offers a distinct approach to executing arithmetic operations and managing data access, including both reading and writing. Therefore, teaching parallel programming through these BLAS routines is not only logical but also advantageous for applications in science, engineering, and artificial intelligence, ultimately providing essential skills in scientific computing.

VI. Implementation of Course Content Strategies

We have introduced slides, detailed text explanations for each chapter, and videos that summarize the chapters through short presentations. Additionally, each chapter, within its specific context, will include a quiz featuring multiple-choice questions. While we cannot claim that this approach is entirely unique, the manner in which we have structured the course content is distinctive, as is our presentation of the MOOC material. We have also drawn inspiration from the online courses offered by EuroCC at NCC Luxembourg [31]. The following items provide further details about how the course content is organized:

- Videos: Each chapter will feature a video ranging from 5 to 10 minutes in duration that elucidates key concepts. We have chosen to maintain concise video lengths to minimize the likelihood of students skipping content or losing engagement. As illustrated in Fig. 5, the layout adopted in the MOOC enhances viewer retention. Shorter videos are effective in piquing curiosity and facilitating clear understanding. Furthermore, these videos will include references to supplementary articles for in-depth exploration of the topics. Choosing shorter videos increases learning capacity, as proven through numerous studies [58–61]. Therefore, we retained short videos for each chapter.

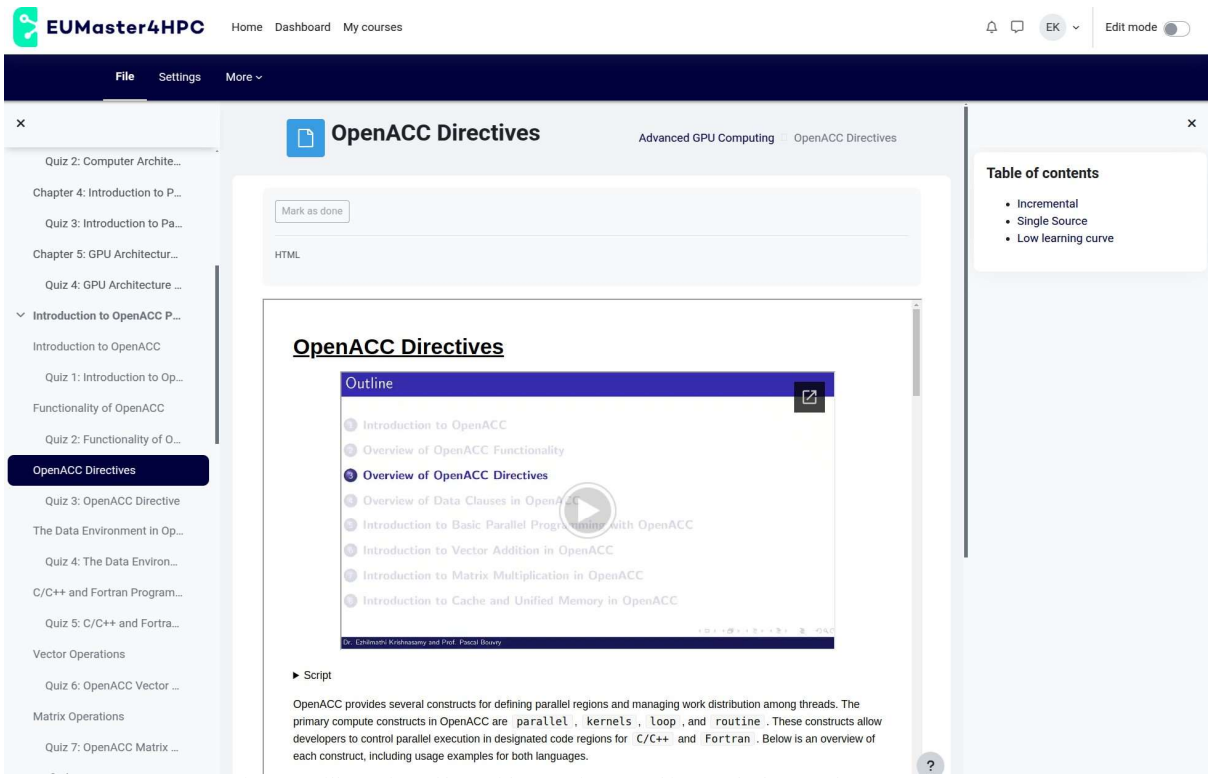

Fig. 5. An illustration of how videos are integrated into each chapter of MOOC content.

- Slides: Each chapter will be accompanied by a dedicated set of slides that underscore critical aspects of the course material. The related videos will provide an elaboration on the information presented within these slides, thereby enriching the learning experience.
- Articles: The articles will provide comprehensive explanations of the subjects covered, supplemented by relevant examples and concluding summaries to enhance clarity and understanding. Given the brevity of the videos, it is essential to provide thorough explanations in a clear and concise manner, similar to the style of a textbook, while avoiding lengthy narratives that include excessive detail. Such detailed coverage is vital, as critical aspects may be overlooked without it, which could potentially result in suboptimal learning outcomes for students. This pedagogical approach is supported by prior studies [62]. An example of an article featured in the MOOC is illustrated in Fig. 6. Furthermore, each article will include a summary of its content, highlighting key concepts that readers should understand upon completing the article; this is illustrated in Fig. 7.

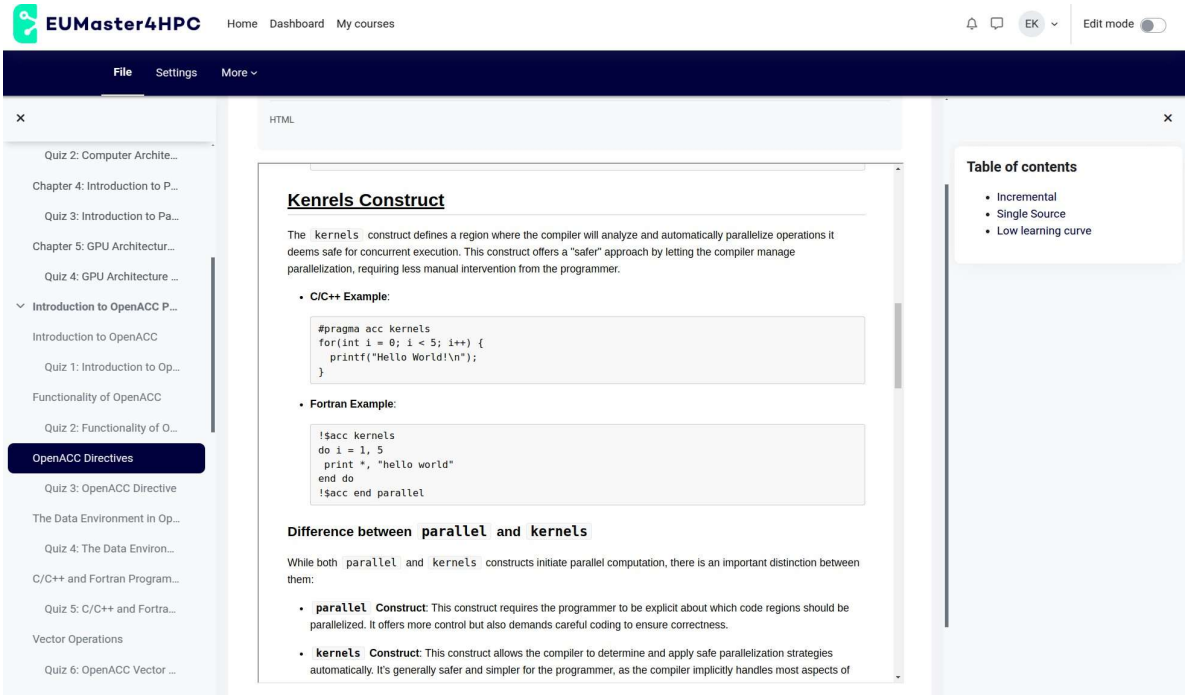

Fig. 6. An example of the article (text) explaining the specific chapter content.

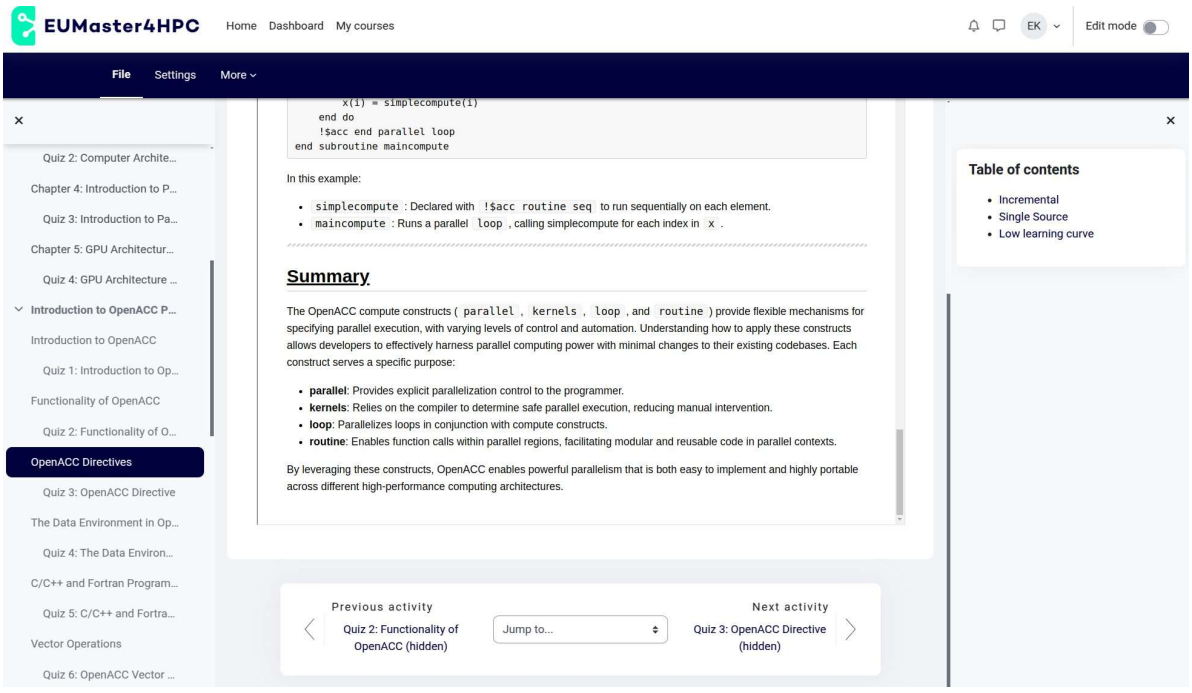

Fig. 7. A summary example of OpenACC compute constructs.

- Quizzes: Each chapter will feature a quiz consisting of more than five questions. These quizzes will incorporate a combination of fill-in-the-blank questions and multiple-choice items, designed to assess students' comprehension of the material, as illustrated in Fig. 8. Incorporating quizzes after each chapter has been

covered has been shown to enhance students' learning capacity. This approach not only ensures that students engage with the video content but also verifies their understanding of the chapter material. It aligns with established pedagogical practices in the realm of MOOCs [63]. Furthermore, our quizzes go beyond merely assessing comprehension of the video content; they also include questions based on the articles provided for each chapter. This comprehensive assessment strategy ensures a holistic evaluation of students' grasp of the course content, ultimately contributing to a deeper understanding of the subject matter.

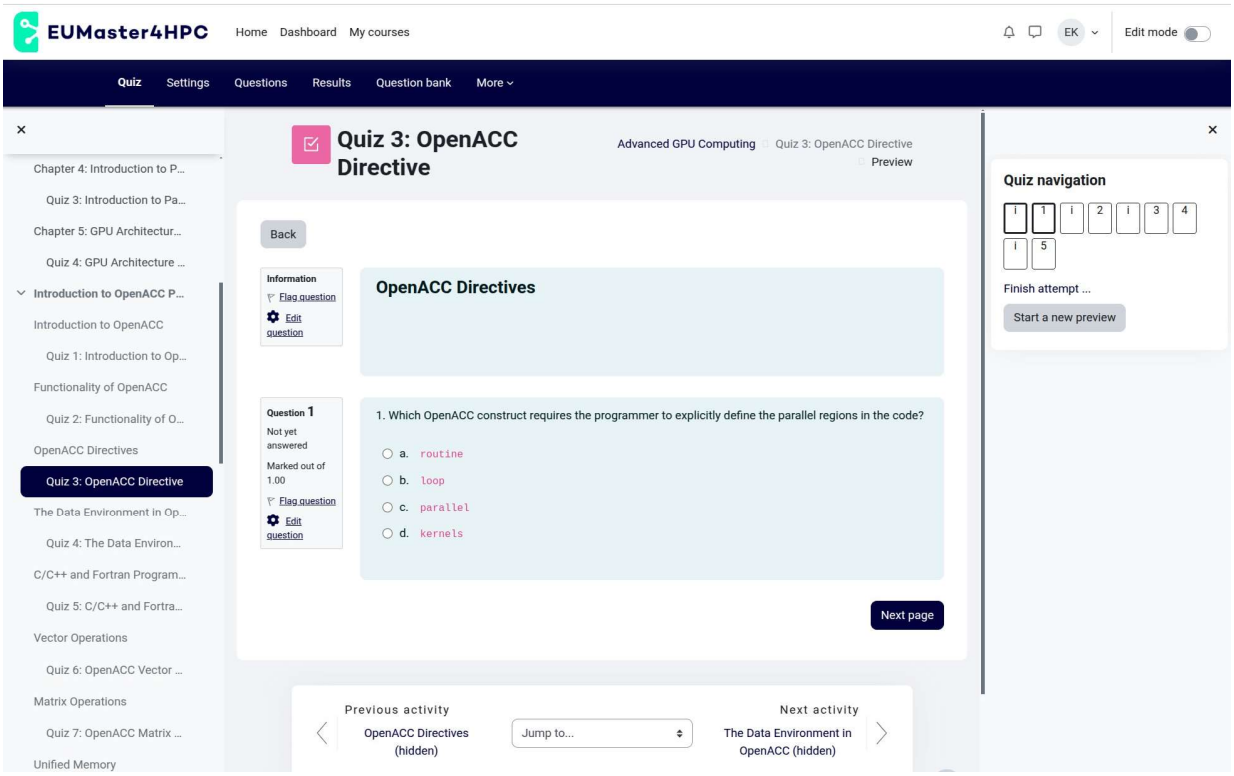

Fig. 8. An example of questions organized within the chapter.

VII. Empirical Study

To assess the course content and determine its alignment with the established learning outcomes, students were given the opportunity to complete a questionnaire upon completing the course. This questionnaire utilized a binary response format, as it facilitates more straightforward analysis of the feedback compared to a multiple-choice format. The table below summarizes the feedback received from 22 students.

From Table 4, it is evident that the course has received a satisfactory rating of up to 79%. However, several areas have been identified for improvement, specifically in the quizzes and examples provided. At this stage, we will refrain from making any immediate changes and instead focus on enhancing the lower-scoring areas in the next iteration of the course. The next iteration is scheduled for launch in September 2025.

Table 4. Evaluation metrics for MOOC effectiveness: student feedback on learning outcomes and course experience.

Evaluation Criteria	Satisfaction %
Does the MOOC align with learning outcomes?	85%
How easy is it to follow the course?	80%
Are the examples well integrated with the concepts?	75%
Does the article provide detailed information?	90%
Do slides and videos deliver high-level content?	82%
Do the quizzes related to the course content?	70%
Overall satisfaction?	79%

VIII. Conclusions and Future Work

This course has received positive feedback regarding the integration of its content, which effectively supports the learning outcomes outlined in the syllabus. In the future, we plan to adopt a similar approach to explore additional course topics, including more advanced subjects such as multiple GPU programming and performance analysis in parallel programming. Currently, this MOOC is only accessible to students from the EUMaster4HPC program; however, we intend to make this content available to a broader audience through another MOOC platform or by opening the EUMaster4HPC MOOC to a wider audience.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Ezhilmathi Krishnasamy: Writing – original draft, review & editing, Conceptualization. Pascal Bouvry: Conceptualization. All authors had approved the final version.

Acknowledgment

This research was supported by the EUMaster4HPC project and received funding from the European High-Performance Computing Joint Undertaking (JU) under Grant Agreement No. 101051997. We would also like to express

our gratitude to the EuroHPC Joint Undertaking for providing access to the HPC (LUMI, Vega, and MeluXina) resources essential for the development of this course content.

REFERENCES

[1] EUMaster4HPC. (2025). *EUMaster4HPC: European Master for High Performance Computing*. [Online]. Available: https://eumaster4hpc.uni.lu/

[2] P. Bouvry *et al*., "The European master for HPC curriculum," *J. Parallel Distrib. Comput*., 105081, 2025.

[3] Udacity, *Parallel Programming*, 2024.

[4] J. H. University, *GPU Programming Specialization*, 2024.

[5] OpenACC. (2025). *OpenACC—Directives for Accelerators*. [Online]. Available: https://www.openacc.org/

[6] OpenMP. (2025). *OpenMP Offloading—Directives for Heterogeneous Computing*. [Online]. Available: https://www.openmp.org/

[7] HIP. (2025). *HIP—A C++ Runtime API for AMD and NVIDIA GPUs*. [Online]. Available: https://github.com/ROCm-Developer-Tools/HIP

[8] Coursera. (2025). *Coursera Plus*. [Online]. Available: https://www.coursera.org/courseraplus

[9] edX. (2025). *edX—Online Courses from the World's Best Universities*. [Online]. Available: https://www.edx.org/

[10] Udacity. (2025). *Udacity—Online Learning Platform for Technology Skills*. [Online]. Available: https://www.udacity.com/

[11] FutureLearn. (2025). *FutureLearn—Online Courses and Degrees*. [Online]. Available: https://www.futurelearn.com/

[12] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming*, Addison-Wesley Professional, 2010.

[13] MIT. (2025). *MIT OpenCourseWare—Free Online Course Materials*. [Online]. Available: https://ocw.mit.edu/

[14] Harvard University. (2025). *Harvard Online Learning—Free and Paid Courses*. [Online]. Available: https://online-learning.harvard.edu/

[15] Stanford University. (2025). *Stanford Online—Free and Paid Courses*. [Online]. Available: https://online.stanford.edu/

[16] University of London. (2025). *University of London Online Courses*. [Online]. Available: https://www.london.ac.uk/courses?mode=online

[17] Open edX. (2025). *Open edX—Open-Source Learning Management System*. [Online]. Available: https://open.edx.org/

[18] Moodle. (2025). *Moodle—Open-Source Learning Platform*. [Online]. Available: https://moodle.org/

[19] Canvas. (2025). *Canvas LMS—Learning Management System*. [Online]. Available: https://www.instructure.com/canvas

[20] Google Classroom. (2025). *Google Classroom - Online Teaching and Learning Platform*. [Online]. Available: https://classroom.google.com/

[21] S. Papadakis, "MOOCs 2012–2022: An overview," *Adv. Mobile Learn. Educ. Res*., vol. 3, no. 1, pp. 682–693, 2023.

[22] M. H. Baturay, "An overview of the world of MOOCs," *Procedia-Soc. Behav. Sci*., vol. 174, pp. 427–433, 2015.

[23] J. R. Drake, M. T. O'Hara, and E. Seeman, "Five principles for MOOC design: With a case study," *J. Inf. Technol. Educ. Innov. Pract*., vol. 14, p. 125, 2015.

[24] A. Margaryan, M. Bianco, and A. Littlejohn, "Instructional quality of massive open online courses (MOOCs)," *Comput. Educ*., vol. 80, pp. 77–83, 2015.

[25] F. Amiti, "Synchronous and asynchronous e-learning," *Eur. J. Open Educ. E-Learn. Stud.*, vol. 5, no. 2, 2020.

[26] R. M. Branch and T. A. Dousay, *Survey of Instructional Development Models*, 2015.

[27] K. L. Heaster-Ekholm, "Popular instructional design models: Their theoretical roots and cultural considerations," *Int. J. Educ. Dev. Using Inf. Commun. Technol*., vol. 16, no. 3, pp. 50–65, 2020.

[28] T. A. Dousay and J. E. Stefaniak, "Instructional design models," in *Found. Learn. Instr. Des. Technol.*, R. E. West, Ed., EdTech Books, 2022.

[29] R. M. Branch, *Instructional Design: The ADDIE Approach*, Springer, 2009.

[30] M. D. Merrill, "First principles of instruction: A synthesis," *Trends Issues Instr. Des. Technol*., vol. 2, pp. 62–71, 2007.

[31] E. Krishnasamy and P. Bouvry, "HPC courses training organization and experiences in Supercomputing Luxembourg EuroCC: National Competence Centre (NCC)," *J. Comput. Sci. Educ*., vol. 15, no. 2, pp. 16–23, 2024.

[32] (2025). GPU programming for scientific computing. *Online Course*. [Online]. Available: https://www.futurelearn.com/courses/gpu-programming-scientific-computing

[33] L. Sanchez, J. Penarreta, and X. S. Poma, "Learning management systems for higher education: A brief comparison," *Discov. Educ*., vol. 3, no. 1, p. 58, 2024.

[34] N. Kajla, K. Naik, and P. Kumar, "Evaluation of open-source learning management systems (LMS) using design science research methodology," in *Proc. Int. Conf. Res. Manag. Eng*., 2021, pp. 56–69.

[35] M. Cone. (2025). Markdown guide. [Online]. Available: https://www.markdownguide.org/

[36] P. R. Lowenthal and C. B. Hodges, "In search of quality: Using quality matters to analyze the quality of Massive, Open, Online Courses (MOOCs)," *Int. Rev. Res. Open Distrib. Learn*., vol. 16, no. 5, pp. 83–101, 2015.

[37] Online Learning Consortium. (2025). *OLC Dimensions of Quality*. [Online]. Available: https://onlinelearningconsortium.org/about/olc-dimensions-of-quality/

[38] Quality Matters. (2025). *Quality Matters: Quality Assurance in Online Learning*. [Online]. Available: https://www.qualitymatters.org/

[39] TOP500. (2025). T*OP500: The List of the World's Fastest Supercomputers*. [Online]. Available: https://top500.org/

[40] Khronos Group. (2025). *OpenCL: The Open Standard for Parallel Programming of Heterogeneous Systems*. [Online]. Available: https://www.khronos.org/opencl/

[41] Intel. (2022). Introduction to the Xe-HPG architecture. *Tech. Rep*. [Online]. Available: https://cdrdv2-public.intel.com/758302/introduction-to-the-xe-hpg-architecture-white-paper.pdf

[42] NVIDIA. (2020). NVIDIA A100 tensor core GPU architecture. *Tech. Rep*. [Online]. Available: https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf

[43] AMD. (2021). Introducing AMD CDNA™ 2 architecture. *Tech. Rep*. [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/instinct-business-docs/white-papers/amd-cdna2-white-paper.pdf

[44] Khronos Group. (2020). *SYCL 2020 Specification*. [Online]. Available: https://www.khronos.org/registry/SYCL/specs/sycl-2020/html/sycl-2020.html

[45] LLNL. (2025). *RAJA User Guide*. [Online]. Available: https://raja.readthedocs.io/en/develop/sphinx/userguide/index.html

[46] P. Pindl, "Performance portability for HPC applications through the RAJA abstraction layer," in *Proc. Int. Conf. High Perform. Comput*., 2022.

[47] 47NVIDIA. (2025). *Thrust: Parallel Algorithms Library*. [Online]. Available: https://developer.nvidia.com/thrust

[48] Kokkos Team. (2025). *Kokkos: Performance Portability for Productive Parallel Programming*. [Online]. Available: https://kokkos.org/

[49] RAJA Team. (2025). *RAJA: Performance Portability for HPC Applications*. [Online]. Available: https://raja.readthedocs.io/en/develop/

[50] NVIDIA. (2025). *NVIDIA Ampere Architecture White Paper*. [Online]. Available: https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf

[51] AMD. (2025). *AMD CDNA2 Architecture White Paper*. [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/instinct-business-docs/white-papers/amd-cdna2-white-paper.pdf

[52] NVIDIA. (2025). *NVIDIA HPC SDK*. [Online]. Available: https://developer.nvidia.com/hpc-sdk

[53] HPE. (2025). *HPE Cray Programming Environment Documentation.* [Online]. Available: https://cpe.ext.hpe.com/docs/latest/cce/index.html

[54] GCC. (2025). *GCC Offloading—Enabling Offload Capabilities in GCC.* [Online]. Available: https://gcc.gnu.org/wiki/Offloading

[55] NVIDIA. (2025). *NVIDIA Nsight Systems.* [Online]. Available: https://developer.nvidia.com/nsight-systems

[56] NVIDIA. (2025). *NVIDIA Nsight Compute*. [Online]. Available: https://developer.nvidia.com/nsight-compute

[57] AMD ROCm Team. (2025). *ROCm Profilers Documentation*. [Online]. Available: https://rocm.blogs.amd.com/software-tools-optimization/profilers/README.html

[58] R. Berg *et al*., "Leveraging recorded mini-lectures to increase student learning," *Online Classroom*, vol. 14, no. 2, pp. 5–8, 2014.

[59] R. Deng and P. Benckendorff, "What are the key themes associated with the positive learning experience in MOOCs? An empirical investigation of learners' ratings and reviews," *Int. J. Educ. Technol. High. Educ.*, vol. 18, no. 1, p. 9, 2021.

[60] N. Abhishek *et al*., "Effectiveness of MOOCs on learning efficiency of students: A perception study," *J. Res. Innov. Teach. Learn*., 2023.

[61] P. J. Guo, J. Kim, and R. Rubin, "How video production affects student engagement: An empirical study of MOOC videos," in *Proc. First ACM Conf. Learn. Scale*, 2014, pp. 41–50.

[62] B. Robertson and M. J. Flowers, "Determining the impact of lecture videos on student outcomes," *Learn. Teach*., vol. 13, no. 2, pp. 25–40, 2020.

[63] G. Kovacs, "Effects of in-video quizzes on MOOC lecture viewing," in *Proc. Third ACM Conf. Learn. Scale*, 2016, pp. 31–40. https://dl.acm.org/doi/10.1145/2876034.2876041

Copyright © 2025 by the authors. This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited (CC BY 4.0).