

Thematic CERN School of Computing - spring 2021

Monday 14 June 2021 - Friday 18 June 2021

Online event

Academic programme

The school will focus on the theme of **Scientific Software for Heterogeneous Architectures**. The complete programme will offer 24 hours of lectures, hands-on exercises and group assignment work, as well as an additional student presentations session, and a special evening lecture.

Introduction lecture

Preparing for the HL-LHC computational challenge

by Danilo Piparo (CERN)

HEP data processing and analysis workflows

Upgrades of the LHC accelerator and experiments

Evolution of hardware and computing infrastructure

Impact on HEP data processing software

Track 1: Technologies and Platforms

4 hours of lectures and 2 hours of group assignment

by Andrzej Nowak

Introduction to efficient computing

The evolution of computing hardware and what it means in practice

The seven dimensions of performance

Controlling and benchmarking your computer and software

Software that scales with the hardware

Advanced performance tuning in hardware

Data-oriented design

Hardware vectorization in detail – theory vs. practice

Software design for vectorization and smooth data flow

How can compilers and other tools help?

Hardware evolution and heterogeneity

Accelerators, co-processors, heterogeneity

Memory architectures, hardware caching and NUMA

Compute devices: CPU, GPU, FPGA, ASIC etc.

The role of compilers

Summary and future technologies overview

Teaching program summary and wrap-up

Next-generation memory technologies and interconnect

Future computing evolution

Track 2: Parallel and Optimised Scientific Software

4 hours of lectures, 1.5 hours of group assignment, and 4 hours of hands-on exercises

by Sebastien Ponce (CERN)

and Danilo Piparo (CERN)

exercises assisted by Arthur Hennequin (CNRS)

Writing parallel software (*D.Piparo*)

Amdahl's and Gustafson's laws

Asynchronous execution

Finding concurrency, task vs. data parallelism

Using threading in C++ and Python, comparison with multi-process

Resource protection and thread safety

Locks, thread local storage, atomic operations

Modern programming languages for HEP (*S.Ponce*)

Why Python and C++ ?

Recent evolutions: C++ 11/14/17

Modern features of C++ related to performance

Templating versus inheritance, pros and cons of virtual inheritance

Python 3, and switching from Python 2

Optimizing existing large codebase (*S.Ponce*)

Measuring performance, tools and key indicators

Improving memory handling

The nightmare of thread safety

Code modernization and low level optimizations

Data structures for efficient computation in modern C++

Practical vectorization (*S.Ponce*)

Measuring vectorization level

What to expect from vectorization

Preparing code for vectorization

Vectorizing techniques in C++: intrinsics, libraries, autovectorization

Track 3: Programming for Heterogeneous Architectures

4 hours of lectures, 1.5 hours of group assignment, and 4 hours of hands-on exercises

by Dorothea vom Bruch (CPPM/CNRS)

and Daniel Campora (University of Maastricht)

Scientific computing on heterogeneous architectures (*D.vom Bruch*)

Introduction to heterogeneous architectures and the performance challenge

From general to specialized: Hardware accelerators and applications

Type of workloads ideal for different accelerators

Trade-offs between multi-core and many-core architectures

Implications of heterogeneous hardware on the design and architecture of scientific software

Embarrassingly parallel scientific applications in HPC and CERN

Programming for GPUs (*D.vom Bruch*)

From SIMD to SPMD, a programming model transition

Thread and memory organization

Basic building blocks of a GPU program

Control flow, synchronization, atomics

Performant programming for GPUs (*D.Campora*)

Data locality, coalesced memory accesses, tiled data processing

GPU streams, pipelined memory transfers

Under the hood: branchless, warps, masked execution

Debugging and profiling a GPU application

Design patterns and best practices (*D.Campora*)

Good practices: single precision, floating point rounding, avoid register spilling, prefer single source

Other standards: SYCL, HIP, OpenCL

Middleware libraries and cross-architecture compatibility

Reusable parallel design patterns with real-life applications

Additional lectures

Student lightning talks session

Special evening lecture

Future of the Universe and of Humanity

by Ivica Puljak (University of Split)