European Organization for Nuclear Research
Organisation européenne pour la recherche nucléaire

CERN School of Computing
CERN – IT Department
CH-1211 Genève 23
Switzerland

Email: computing.school@cern.ch

Georgios Voulgarakis
CERN

Our reference: tCSC2017-certif

Geneva, 13 June 2017

TO WHOM IT MAY CONCERN

This is to certify that:

Georgios Voulgarakis

attended the Thematic CERN School of Computing 2017 held at MEDILS in Split, Croatia - organised in collaboration with the University of Split - from 4-10th June 2017.

The academic programme of the School consisted of 12 hours of lectures (plus 2 special guest lectures), and 10 hours of exercises. A summary of the syllabus is overleaf.

The School website can be found at https://indico.cern.ch/event/591480/

Sebastian Lopienski

Sebastian Lopienski
Director
CERN School of Computing

CERN
School of Computing

Thematic CERN School of Computing 2017

Efficient Parallel Processing of Future Scientific Data

Introduction

- Future scientific data processing: challenges in HEP and other sciences, commonalities and differences.
- The prime role of software in modern big science.
- Parallelism and asynchronism: computation and I/O.
- Evolution of hardware and platforms, consequences on data analysis procedures and tools.

Track 1: Technologies and Platforms

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

Intermediate Concepts in Efficient Computing

- Memory architectures, hardware caching and NUMA
- Scaling out: Big Data – Big Hardware
- The role of compilers and VMs
- A brief look at accelerators and heterogeneity

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?

Summary and Future Technologies Overview

- Teaching program summary and wrap-up
- Next-generation memory technologies and interconnect
- Rack-sized datacenters and future computing evolution
- Software technologies – forecasts

Track 2: Programming for concurrency and correctness

Scientific software programming: a modern approach

- Introduction: Amdahl's law, Performance and correctness of codebases
- Modern C++: new constructs, their advantages
- Exploit modern architectures using Python
- Near the hardware: the role of compilers
- Understanding the differences and commonalities of data structures, metrics for their classification, concrete examples

Expressing Parallelism Pragmatically

- Trivial asynchronous execution
- Task and data decomposition
- Threads and the thread pool model
- In depth comparison of threads and processes, guidelines to choose the best option

Protection of Resources and Thread Safety

- The problem of synchronization
- Useful design principles
- Replication, atomics, transactions and locks
- Lock-free programming techniques
- Functional programming style and elements of map-reduce
- Third party libraries and high level solutions

Ensure Correctness of a Parallel Scientific Application

- Correctness and reproducibility of a scientific result
- Stability of results and testing: regression, physics performance, tradeoffs
- Enforce avoiding thread unsafe constructs: focus on static analysis
- Algorithms for detecting synchronisation pathologies: focus on the DRD and Helgrind tools
- Elements of the GNU debugger: introduction and specific usage in the multithreaded case

---

Track 3: Effective I/O for Scientific Applications

Structuring data for efficient I/O

- Pro/cons of row-column and mixed formats
- compression and its efficiency dependencies on variable types, impact of data format
- Data addressing : limitation of hierarchical approach, usage of flat namespaces
- Stateful vs stateless interfaces for namespaces and I/O

Many ways to store data

- Storage devices and their specificities
- Data federation
- Parallelizing files storage
- Introduction to the Map/Reduce pattern

Preserving Data

- Risks of data loss and corruption
- Data consistency (checksumming)
- Data safety (redundancy, parity, erasure coding)

Key Ingredients to achieve effective I/O

- Asynchronous I/O
- I/O optimizations
- Caching
- Influence of data structures on I/O efficiency