# inverted CERN School of Computing 2016

# Report of Contributions

https://indico.cern.ch/e/iCSC2016

Contribution ID: **0** Type: **not specified**

# I - Event reconstruction in Modern Particle Physics

*Tuesday 1 March 2016 09:00 (1 hour)*

Particle physics experiments have always been at the forefront of big data experiments: the upgrade of the LHCb experiment will lead to data rates greater than 10Tb's per second! This is key to the success of high-energy physics, where large data samples, sophisticated triggers and robust simulations have lead to observing and understanding extremely rare events, including the Higgs Boson. Continuously, physicists are revisiting computing and electronics decisions to balance the differences between the quality and quantity of physics results, computing effort and available budgets. By drawing on examples of modern particle physics experiments, these lectures will consider the various approaches to tackle such large particle physics data problems:

• Data reduction at the hardware level, including triggers.
• Principles and optimizations of reconstruction algorithms.
• Parallelized reconstruction, including sub processing vs. multithreading.

These topics will be introduced in order, beginning with raw data output from these large experiments, passing through the different stages of data reconstruction and reduction, leading to examples of physics results. The lecture series will end with an outlook to future technologies and the associated physics.

**Presenter:** SAUNDERS, Daniel Martin (University of Bristol (GB))

Contribution ID: **1** Type: **not specified**

# I - Detector Simulation for the LHC and beyond: how to match computing resources and physics requirements

*Tuesday 1 March 2016 10:00 (1 hour)*

Detector simulation at the LHC is one of the most computing intensive activities. In these lectures we will show how physics requirements were met for the LHC experiments and extrapolate to future experiments (FCC-hh case).

At the LHC, detectors are complex, very precise and ambitious: this implies modern modelisation tools for geometry and response. Events are busy and characterised by an unprecedented energy scale with hundreds of particles to be traced and high energy showers to be accurately simulated. Furthermore, high luminosities imply many events in a bunch crossing and many bunch crossings to be considered at the same time. In addition, backgrounds not directly correlated to bunch crossings have also to be taken into account.

Solutions chosen for ATLAS (a mixture of detailed simulation and fast simulation/parameterisation) will be described and CPU and memory figures will be given. An extrapolation to the FCC-hh case will be tried by taking as example the calorimeter simulation.

**Presenter:** CAIRO, Valentina (Universita della Calabria (IT))

Contribution ID: **2** Type: **not specified**

# II - Template Metaprogramming for Massively Parallel Scientific Computing - Vectorization with Expression Templates

*Tuesday 1 March 2016 11:30 (1 hour)*

Large scale scientific computing raises questions on different levels ranging from the fomulation of the problems to the choice of the best algorithms and their implementation for a specific platform. There are similarities in these different topics that can be exploited by modern-style C++ template metaprogramming techniques to produce readable, maintainable and generic code. Traditional low-level code tend to be fast but platform-dependent, and it obfuscates the meaning of the algorithm. On the other hand, object-oriented approach is nice to read, but may come with an inherent performance penalty.

**These lectures aim to present he basics of the Expression Template (ET) idiom which allows us to keep the object-oriented approach without sacrificing performance. We will in particular show to to enhance ET to include SIMD vectorization. We will then introduce techniques for abstracting iteration, and introduce thread-level parallelism for use in heavy data-centric loads. We will show to to apply these methods in a way which keeps the "front end" code very readable.**

LECTURE 2
In this lecture, we will have a closer look at the opportunities for implementing SIMD vectorisation through the Expression Template idiom. We will see how it can create a layer of separation between the algorithm, and the low-level implementation. We will use the C++ template mechanisms to accommodate our program so that the algorithm itself doesn't need to explicitly specify SIMD-related types alignment, or operations. We will also explore how our memory data structure layout affects SIMD performance in different workloads, and introduce methods which improve performance in specific cases.

**Presenter:** VYSKOČIL, Jiří (Czech Technical University in Prague)

Contribution ID: **3** Type: **not specified**

# II - Multivariate Classification and Machine Learning in HEP

*Tuesday 1 March 2016 14:00 (1 hour)*

A summary of the history of deep-learning is given and the difference to traditional artificial neural networks is discussed.
Advanced methods like convoluted neural networks, recurrent neural networks and unsupervised training are introduced.
Interesting examples from this emerging field outside HEP are presented. Possible applications in HEP are discussed.

**Presenter:** KECK, Thomas (KIT)

Contribution ID: **4** Type: **not specified**

# Formal Verification - Robust and Efficient code: Why Formal Verification

*Tuesday 1 March 2016 15:00 (1 hour)*

<p>LECTURE 2: In this lecture we will expand on the concepts of the previous lecture and establish formal methods in a broader context, ignoring implementation detail, and investigate how and where these methods are used today, and where they might be used tomorrow. As concrete examples we will be studying how FV can benefit static analysis and comp-cert, and verified C compiler.</p>

<hr>

<p>This talk aims to introduce the concepts of Formal Verification and how they can be used to the benefit of the programmer to produce robust and efficient code. We will be looking into the subject at two levels, both and overview of what FV can concretely bring programmers and going into the nitty-gritty details of theorem proving one of the methods use for FV.</p>

<p>In general, FV means ”proving that certain properties hold for a given system using formal mathematics”. This definition can certainly feel daunting, however, as we will learn, we can reap benefits from the paradigm without digging too deep into the subject.</p>

<p>Examples where FV can help include proving that your code cannot raise division by zero exceptions; produce optimised byte code where the optimisations are proven to be safe and help reason about concurrent systems.</p>

**Presenter:** ALBERTSSON, Kim (CERN)

Contribution ID: **5** Type: **not specified**

# Accelerating C++ applications in Medical Physics

*Tuesday 1 March 2016 16:30 (1 hour)*

The recent developments in multithreading tools in C++, like OpenMP and TBB, taking advantage of the multicore architecture of the nowadays processors, allowed the creation and improvement of powerful softwares for scientific research. This talk will be focused on the development of such software for simulations, data acquisition and image reconstruction in Positron Emission Tomography, one of the most powerful tools for cancer detection.

**Presenter:** MENDES CORREIA, Pedro Manuel (University of Aveiro (PT))

Contribution ID: **6** Type: **not specified**

# II - Event reconstruction in Modern Particle Physics

*Wednesday 2 March 2016 09:00 (1 hour)*

Particle physics experiments have always been at the forefront of big data experiments: the upgrade of the LHCb experiment will lead to data rates greater than 10Tb's per second! This is key to the success of high-energy physics, where large data samples, sophisticated triggers and robust simulations have lead to observing and understanding extremely rare events, including the Higgs Boson. Continuously, physicists are revisiting computing and electronics decisions to balance the differences between the quality and quantity of physics results, computing effort and available budgets. By drawing on examples of modern particle physics experiments, these lectures will consider the various approaches to tackle such large particle physics data problems:

• Data reduction at the hardware level, including triggers.
• Principles and optimizations of reconstruction algorithms.
• Parallelized reconstruction, including sub processing vs. multithreading.

These topics will be introduced in order, beginning with raw data output from these large experiments, passing through the different stages of data reconstruction and reduction, leading to examples of physics results. The lecture series will end with an outlook to future technologies and the associated physics.

**Presenter:** SAUNDERS, Daniel Martin (University of Bristol (GB))

Contribution ID: **7** Type: **not specified**

# II - Detector simulation for the LHC and beyond : how to match computing resources and physics requirements

*Wednesday 2 March 2016 10:00 (1 hour)*

Detector simulation at the LHC is one of the most computing intensive activities. In these lectures we will show how physics requirements were met for the LHC experiments and extrapolate to future experiments (FCC-hh case).

At the LHC, detectors are complex, very precise and ambitious: this implies modern modelisation tools for geometry and response. Events are busy and characterised by an unprecedented energy scale with hundreds of particles to be traced and high energy showers to be accurately simulated. Furthermore, high luminosities imply many events in a bunch crossing and many bunch crossings to be considered at the same time. In addition, backgrounds not directly correlated to bunch crossings have also to be taken into account.

Solutions chosen for ATLAS (a mixture of detailed simulation and fast simulation/parameterisation) will be described and CPU and memory figures will be given. An extrapolation to the FCC-hh case will be tried by taking as example the calorimeter simulation.

**Presenter:** CAIRO, Valentina (Universita della Calabria (IT))

Contribution ID: **8** Type: **not specified**

# III - Template Metaprogramming for massively parallel scientific computing - Templates for Iteration; Thread-level Parallelism

*Wednesday 2 March 2016 11:30 (1 hour)*

Large scale scientific computing raises questions on different levels ranging from the fomulation of the problems to the choice of the best algorithms and their implementation for a specific platform. There are similarities in these different topics that can be exploited by modern-style C++ template metaprogramming techniques to produce readable, maintainable and generic code. Traditional low-level code tend to be fast but platform-dependent, and it obfuscates the meaning of the algorithm. On the other hand, object-oriented approach is nice to read, but may come with an inherent performance penalty.

**These lectures aim to present he basics of the Expression Template (ET) idiom which allows us to keep the object-oriented approach without sacrificing performance. We will in particular show to to enhance ET to include SIMD vectorization. We will then introduce techniques for abstracting iteration, and introduce thread-level parallelism for use in heavy data-centric loads. We will show to to apply these methods in a way which keeps the "front end" code very readable.**

LECTURE 3
In this lecture, we will look into a specific technique to parallelize a large data-centric workload iterating over a multi-dimensional array. We will show how to separate iteration and computation and how the "front-end" algorithm can then be made independent on the dimensionality, coordinate system, or order of numerical approximation. We will show how this separation further helps to implement thread-level parallelism into the "back-end" and explore some common cases of data dependency. We will finally take a look at an example code combining the ideas of all three lectures.

**Presenter:** VYSKOČIL, Jiří (Czech Technical University in Prague)

Contribution ID: **9** Type: **not specified**

# Shared memory and message passing revisited in the many-core era

*Wednesday 2 March 2016 14:00 (1 hour)*

In the 70s, Edsgar Dijkstra, Per Brinch Hansen and C.A.R Hoare introduced the fundamental concepts for concurrent computing. It was clear that concrete communication mechanisms were required in order to achieve effective concurrency.

Whether you're developing a multithreaded program running on a single node, or a distributed system spanning over hundreds of thousands cores, the choice of the communication mechanism for your system must be done intelligently because of the implicit programmability, performance and scalability trade-offs. With the emergence of many-core computing architectures many assumptions may not be true anymore.

In this talk we will try to provide insight on the characteristics of these communication models by providing basic theoretical background and then focus on concrete practical examples based on indicative use case scenarios. The case studies of this presentation cover popular programming models, operating systems and concurrency frameworks in the context of many-core processors.

**Presenter:** SANTOGIDIS, Aram (CERN)

Contribution ID: **10** Type: **not specified**

# Volatile Environments with Virtualisation Technologies

*Wednesday 2 March 2016 15:00 (1 hour)*

Sometimes our job or even our interest to learn something new, requires from us to install a lot of different software to allow a specific program to run on our operating system. This (in the best case) might just prohibit your program to run due to conflicts between different library or language versions; in the worst case your operating system will start becoming full of junk and later on will be slow and insecure. In this course we will explore two relatively new but very well established virtualisation technologies: Vagrant and Docker and how those tools can help us to keep a tidy, exportable and transferable developing environment on our home or work computer.

**Presenter:** ANDRONIDIS, Anastasios (University of Ioannina (GR))

Contribution ID: **11** Type: **not specified**

# Continuous Integration : how can it help?

*Monday 29 February 2016 09:30 (1 hour)*

<p>Software becomes more complex as the project size and number of developers grow. As these two factors increase, so too does the potential for more errors in the code. Previous time and money can be wasted trying to track down bugs that could have been easily avoided should there have been a good workflow in place.</p>

<p>Continuous Integration (CI) is one such strategy that can dramatically improve software quality. An in-depth look at what CI is, as well as the fundamental concepts will be explored. Various scenarios of how CI can be incorporated into different types of projects will be covered. There are many CI software packages on the market. It's not always easy choosing what CI package is best suited for your project. Some main points to keep in mind when beginning to implement CI into your project will also be discussed.</p>

**Presenter:** SMITH, Joshua Wyatt (Georg-August-Universitaet Goettingen (DE))

Contribution ID: **12** Type: **not specified**

# Continuous Delivery and Quality Monitoring

*Monday 29 February 2016 11:00 (1 hour)*

We're all involved in some software/physics projects. As a rule of thumb projects start really simple - a couple of scripts, classes and a few external dependencies. At this phase delivering a release to our clients is simple. We can compile the project locally and deliver compiled sources, for example by e-mail. Unfortunately, in most cases the growth of projects is inevitable. Our simple approaches to build, test and deliver applications are not sufficient. We start to spend more and more time on these 'administrative'procedures than on the real developments. As the project grows, our productivity declines and we are less responsive to requests from our clients.
In this lecture I will try to present common delivery patterns and tools which facilitate these processes.
After introducing Continuous Delivery, I will switch the topic and try to answer the question how much should we invest in quality and how to do it efficiently. My observations reveal that software quality is often considered as the slowing down force. Following this false belief I would like to convince people that software quality can accelerate development within our projects.

**Presenter:** KROL, Kamil Henryk (CERN)

Contribution ID: **13** Type: **not specified**

# I - Template Metaprogramming for Massively Parallel Scientific Computing - Expression Templates

*Monday 29 February 2016 13:45 (1 hour)*

Large scale scientific computing raises questions on different levels ranging from the fomulation of the problems to the choice of the best algorithms and their implementation for a specific platform. There are similarities in these different topics that can be exploited by modern-style C++ template metaprogramming techniques to produce readable, maintainable and generic code. Traditional low-level code tend to be fast but platform-dependent, and it obfuscates the meaning of the algorithm. On the other hand, object-oriented approach is nice to read, but may come with an inherent performance penalty.

**These lectures aim to present he basics of the Expression Template (ET) idiom which allows us to keep the object-oriented approach without sacrificing performance. We will in particular show to to enhance ET to include SIMD vectorization. We will then introduce techniques for abstracting iteration, and introduce thread-level parallelism for use in heavy data-centric loads. We will show to to apply these methods in a way which keeps the "front end" code very readable.**

LECTURE 1
In this lecture, we will have a quick look at the basics of Template Metaprogramming - how does the computing handle "types", and how these types map more or less naturally to physical quantities. We will then introduce the idea of using Expression Templates (ET) as a means to bride the gap between the low-level high-performance approach, and the object-oriented, readable but often severely under-performing approach. We will study the structure of the ETs, and show the basic steps needed to build them.

**Presenter:** VYSKOČIL, Jiří (Czech Technical University in Prague)

Contribution ID: **14** Type: **not specified**

# I - Multivariate Classification and Machine Learning in HEP

*Monday 29 February 2016 14:45 (1 hour)*

Traditional multivariate methods for classification (Stochastic Gradient Boosted Decision Trees and Multi-Layer Perceptrons) are explained in theory and practise using examples from HEP.
General aspects of multivariate classification are discussed, in particular different regularisation techniques.
Afterwards, data-driven techniques are introduced and compared to MC-based methods.

**Presenter:** KECK, Thomas (KIT)

Contribution ID: **15** Type: **not specified**

# Formal verification - Robust and efficient code: Introduction to Formal Verification

*Monday 29 February 2016 16:15 (1 hour)*

**LECTURE 1: We will establish two general approaches to FV and where they are applicable: model checking and theorem proving. We will explore the latter in more details and have a brief look at the underlying theory, predicate logic. We will see how this family of logic systems can be used to prove abstract properties of our program and why this is useful. Practical examples will be presented and explained.**

This talk aims to introduce the concepts of Formal Verification and how they can be used to the benefit of the programmer to produce robust and efficient code. We will be looking into the subject at two levels, both and overview of what FV can concretely bring programmers and going into the nitty-gritty details of theorem proving one of the methods use for FV.
In general, FV means "proving that certain properties hold for a given system using formal mathematics". This definition can certainly feel daunting, however, as we will learn, we can reap benefits from the paradigm without digging too deep into the subject.
Examples where FV can help include proving that your code cannot raise division by zero exceptions; produce optimised byte code where the optimisations are proven to be safe and help reason about concurrent systems.

**Presenter:** ALBERTSSON, Kim (CERN)

Contribution ID: **16** Type: **not specified**

# A word from the IT Department Head

*Monday 29 February 2016 13:30 (15 minutes)*

**Presenter:** HEMMER, Frederic (CERN)

Contribution ID: **17** Type: **not specified**

# Closing remarks

*Wednesday 2 March 2016 16:00 (15 minutes)*

**Presenter:** PACE, Alberto (CERN)