Welcome to **iCSC2014** the seventh edition of the Inverted School, “*Where Students turn into Teachers*”.

At regular CERN Schools of Computing, the sum of the knowledge of the students often exceeds the one of lecturer teaching, and that it is frequent to find in the room real experts on particular topics. This is the idea behind **iCSCs**.

During the main school, students give lightning talks and make proposals via an electronic discussion forum, from which a programme for the inverted school is inspired.

This year’s programme relies from contribution selected from the main school in Nicosia, Cyprus in August 2013. It focuses challenging and innovative topics: “LAN programming for building distributed systems”, “Web API”, “Read-out electronics: where data come from”, “Machine learning and data mining” and a topic to bridge physics with computing on “A journey from quark to jet”.

We are indebted to the lecturers at the main CSC who acted as mentors. My main thanks go to all those who developed ideas and proposals and to those actually lecturing. This is their school and I am confident all will go very well. It is already the seventh edition and do not hesitate, you the attendees, to comment and advise us on how to improve it.

Enjoy the school.
Contents

Foreword 1
History of iCSCs 5
Schedule 7
iCSC 2014 Lecturers 9
iCSC 2014 Mentors 11
Introduction to the inverted CSC 13

Lecture 1 LAN Programming - The basics 15
Lecture 2 Is your web API truly RESTful (and does it matter) 25
Lecture 3 Building highly distributed systems within 5 minutes 39

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Lecture 4 From Quark to Jet: A Beautiful Journey
Beauty physics, tracking and large-scale distributed computing in HEP 53
Lecture 5 Read-Out Electronics: where data come from 65
Lecture 6 From Quark to Jet: A Beautiful Journey
Making a jet, classifying a jet, and personal scale computing in HEP 77
Lecture 7 Read-Out Electronics: where data come from 85
Lecture 8 Introduction to machine learning and data mining 95
Lecture 9 Self organizing maps. A visualization technique with data dimension reduction 107
# History of iCSCs

| iCSC2005 | 23-25 February 2005, CERN | ✓ Data Management and Database Technologies  
 ✓ Advanced Software Development & Engineering  
 ✓ Web Services in Distributed Computing |
|---|---|---|
| iCSC2006 | 6-8 March 2006, CERN | ✓ Computational Intelligence for HEP Data Analysis  
 ✓ The Art of Designing Parallel Applications  
 ✓ Software Testing: Fundamentals and Best Practices |
| iCSC2008 | 3-5 March 2008, CERN | ✓ Towards Reconfigurable High-Performance Computing  
 ✓ Special topics |
| iCSC2010 | 8-9 March 2010, CERN | ✓ Software management and optimization  
 ✓ Visualisation  
 ✓ System monitoring |
| iCSC2011 | 3-4 March 2011, CERN | ✓ Virtualization and clouds  
 ✓ Cryptography  
 ✓ Modern software engineering |
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>iCSC2013</td>
<td>25-26 February 2013, CERN</td>
</tr>
<tr>
<td>✓ GPU computing and its</td>
<td>applications in HEP</td>
</tr>
<tr>
<td>✓ Introduction to Computer</td>
<td>Vision</td>
</tr>
<tr>
<td>✓ Testing methods and tools for</td>
<td>large scale distributed systems</td>
</tr>
<tr>
<td>✓ How the LHC experiment have</td>
<td>interpreted the Grid distributed computing model</td>
</tr>
<tr>
<td>Time</td>
<td>Lecture</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
</tr>
<tr>
<td>14:00-14:10</td>
<td></td>
</tr>
<tr>
<td>14:10-14:20</td>
<td></td>
</tr>
<tr>
<td>14:20-15:20</td>
<td>Lecture 1</td>
</tr>
<tr>
<td>15:20-15:50</td>
<td></td>
</tr>
<tr>
<td>15:50-16:50</td>
<td>Lecture 2</td>
</tr>
<tr>
<td>16:50-17:50</td>
<td>Lecture 3</td>
</tr>
</tbody>
</table>

**Tuesday 25 February**

<table>
<thead>
<tr>
<th>Time</th>
<th>Lecture</th>
<th>Title</th>
<th>Speaker</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00-10:00</td>
<td>Lecture 4</td>
<td>From Quark to Jet: A Beautiful Journey – Lecture 1</td>
<td>Tyler Mc Millan Dorland</td>
</tr>
<tr>
<td>10:00-11:00</td>
<td>Lecture 5</td>
<td>Read-Out Electronics: where data come from – Lecture 1</td>
<td>Francesco Messi</td>
</tr>
<tr>
<td>11:00-11:30</td>
<td></td>
<td>Coffee break</td>
<td></td>
</tr>
<tr>
<td>11:30-12:30</td>
<td>Lecture 6</td>
<td>From Quark to Jet: A Beautiful Journey – Lecture 2</td>
<td>Tyler Mc Millan Dorland</td>
</tr>
<tr>
<td>12:30-13:30</td>
<td></td>
<td>Lunch</td>
<td></td>
</tr>
<tr>
<td>13:30-14:30</td>
<td>Lecture 7</td>
<td>Read-Out Electronics: where data come from – Lecture 2</td>
<td>Francesco Messi</td>
</tr>
<tr>
<td>14:30-15:30</td>
<td>Lecture 8</td>
<td>Introduction to machine learning and data mining</td>
<td>Juan Lopez Gonzalez</td>
</tr>
<tr>
<td>15:30-16:00</td>
<td></td>
<td>Coffee break</td>
<td></td>
</tr>
</tbody>
</table>
| 16:00-17:00  | Lecture 9| Self organizing maps                                                 | Juan Lopez Gonzalez      
|              |         | A visualization technique with data dimension reduction              |                          |
### DORLAND Tyler  
**DESY, Hamburg - Germany**

My first exposure to working with computers came from blowing into the cartridge of my Nintendo back in 1989. A few years later I had upgraded and was learning the intricacies of my 486 I spent all summer saving for. Since then, I have fiddled around with careers in music, football, and ultimate, but I have always had a computer at my side. I graduated from the University of Colorado with degrees in physics and saxophone performance (BA,BM). I continued my studies in physics at the University of Notre Dame (M.Sc.), and finished them at the University of Washington where I defended my search for the associated production of a vector and Higgs Boson, research conducted as a member of the D-Zero collaboration, as my doctoral thesis. Now I am a fellow at DESY working on the CMS experiment in Top quark studies and hardware upgrades. As always, my trusty computer aids me at every step, I am looking forward to learning more about it!

---

### HAMMER Josef  
**CERN, Geneva - Switzerland**

Having studied computer science in Austria and Australia (with a focus on eCommerce and mobile and distributed systems), I obtained my MSc in 2007. Following nine months of traveling around the world, I zeroed in again on computer networks with a project on automatic Quality of Service management for home networks.

In 2009 I joined CMS as a PhD student and created the SOAP-based Interconnection Test Framework for the distributed CMS L1-Trigger environment. Since 2012 I am working as a fellow for the PH-CMG-CO group, moving our monolithic JSP/Servlet-based system to RESTful web applications.

I have been involved in the design of distributed systems for more than a decade, using a variety of technologies and tools ranging from C++ and CORBA over Zeroconf, Java EE and SOAP-based Web Services to REST and Python.

---

### KUNZE Jonas  
**Johannes Gutenberg Universität Mainz - Germany**

I am a German physicist and PhD student working for the NA62 experiment at CERN and as Linux administrator at the University of Mainz, Germany. During my diploma thesis I’ve developed the PC-farm and implemented the associated software framework for the online trigger of NA62. Through this work and private projects I gained a lot of experience in network and parallel programming.

Within my current work I’m analyzing the feasibility of using commodity processors for low-level triggers at high-energy physics experiments like NA62, LHCb and ATLAS. Privately I have developed the social network [www.metalcon.de](http://www.metalcon.de) which has become the biggest German database concerning heavy metal music. As a part of the current redesign I’ve developed a fast autocompletion service which will run as a parallel web service distributed all over the world.
LÓPEZ GONZÁLEZ Juan  CERN, Geneva - Switzerland

I work for GS-AIS-GDI in development and maintenance of the applications of the group (Phonebook, public outreach, e-groups, roles, foundation data...) I am also responsible of the maintenance and administration of the deployment/build/monitoring infrastructure in GDI section. Programming languages: C++, C#, Java, Groovy (Grails) and Lisp. Others: Maven, Bamboo, Nexus, Atlassian tools.

MESSI Francesco  Rheinische Friedrich-Wilhelms-Universität, Bonn - Germany

I graduated in Physics working on the time-adjustment of the Front-End electronics for the muon chambers of the LHCb experiment in the electronics group of the university of Rome "La Sapienza". I worked one year in the didactic group of the university of Rome "Tor Vergata" developing didactical instrumentation for the high school. I started my PhD in 2009 at the "Rheinische Friedrich-Wilhelms-Universität" in Bonn on the Tagger Detector of the BGO-OD experiment at the ELSA accelerator. The main focus of my work is the development of the Front-End electronics (Amplifier, Dual Threshold Discriminator and Shaper). This includes the design of the PCB, the development of the firmware for the microcontroller and of the User Interface of the board, the commissioning of the boards and the analysis for the in-beam characterization of the prototype detector. I also contributed to the construction of the detector and to the implementation of the analysis tools. At present I am writing my thesis.
### iCSC2014 Mentors

The preparation of each iCSC lecture has been followed by one or two CSC lecturer(s) acting as mentor(s).

<table>
<thead>
<tr>
<th>Mentor</th>
<th>Lecture</th>
<th>Lecturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Francois Fluckiger</td>
<td>LAN Programming – The basics</td>
<td>Jonas Kunze</td>
</tr>
<tr>
<td></td>
<td>Building highly distributed systems within 5 minutes</td>
<td></td>
</tr>
<tr>
<td>Sebastian Lopienski</td>
<td>Is your web API truly RESTful (and does it matter)</td>
<td>Joseph Hammer</td>
</tr>
<tr>
<td>Giulio Lo Presti</td>
<td>Is your web API truly RESTful (and does it matter)</td>
<td>Joseph Hammer</td>
</tr>
<tr>
<td>Pere Mato Vila</td>
<td>Read-Out Electronics: where data come from</td>
<td>Francesco Messi</td>
</tr>
<tr>
<td>Lorenzo Moneta</td>
<td>Introduction to machine learning and data mining</td>
<td>Juan Lopez Gonzalez</td>
</tr>
<tr>
<td></td>
<td>Self organizing maps – A visualization technique with data dimension reduction</td>
<td></td>
</tr>
<tr>
<td>Andrzej Nowak</td>
<td>Introduction to machine learning and data mining</td>
<td>Juan Lopez Gonzalez</td>
</tr>
<tr>
<td></td>
<td>Self organizing maps – A visualization technique with data dimension reduction</td>
<td></td>
</tr>
<tr>
<td>Alberto Pace</td>
<td>Read-Out Electronics: where data come from</td>
<td>Francesco Messi</td>
</tr>
<tr>
<td>Ivica Puljak</td>
<td>From Quark to Jet: A Beautiful Journey</td>
<td>Tyler Mc Millan Dorland</td>
</tr>
<tr>
<td>Are Strandlie</td>
<td>From Quark to Jet: A Beautiful Journey</td>
<td>Tyler Mc Millan Dorland</td>
</tr>
</tbody>
</table>
Welcome …

iCSC

Inverted CERN School of Computing, 24-25 February 2014

The CERN School of Computing

- Aims at creating a common culture in scientific computing among young scientists and engineers involved in particle physics or other sciences, as a strategic direction to favor mobility and to facilitate the development of large computing-oriented transnational projects.
  - http://cern.ch/csc

- Participants come from worldwide laboratories and universities with typically of 15 to 30 different nationalities (60 different nationalities over the past 10 years).
  - http://cern.ch/csc/alumni

The inverted CSC

- At the end of each main school, we call students present to make proposals. When we receive sufficient proposals of appropriate quality, we organize an inverted school.

- The students combine their skills and elaborate on CSC related subjects.

- At regular CSCs, the sum of the knowledge of the students often exceeds the one of lecturer teaching, and that it is frequent to find in the room real experts on particular topics. This is the idea behind iCSC.
This year programme

- **Topics**
  - LAN Programming, building distributed systems
  - Web API
  - A journey from quark to jet
  - Read-Out Electronics: where data come from
  - Machine learning and data mining
- **Speakers from CSC 2013, Nicosia**
  - Francesco, Jonas, Josef, Juan Lopez, and Tyler
- **Printed booklet available to registered participants**

---

**Be just and fear not**

*Shakespeare*
Distributed computing enables you to implement scalable systems with a higher total *computing power* while it also increases the *availability*. As the components of a distributed system have to communicate via networks, it is important to understand the underlying protocols to use them efficiently. In this lecture we will go into details about (10) *Gigabit Ethernet* as this is still the most common LAN technology. In this context the most important Internet Protocols will be discussed and compared.

As nowadays networking performance is typically limited by the operating system, we will also discuss how Internet sockets are implemented in Linux and how alternatives based on Ethernet and other technologies look like.

**Audience**
The attendees will understand the capabilities of the main Internet Protocols and how they perform on a Local Area Network. They will also get to know high performance network drivers significantly increasing throughput and latency.

This lecture targets anyone who is interested in network programming and wants to understand how the underlying architectures work and which are the most common bottlenecks.

**Pre-requisite**
A basic programming knowledge is recommended to follow this lecture.
Network Programming
Lecture 1
LAN Programming – The Basics

Jonas Kunze
University of Mainz – NA62

Outline

- Recap of the TCP/IP model
  - ISO/OSI and TCP/IP
  - User Datagram Protocol (UDP)
  - Transmission Control Protocol (TCP)
- Network programming with BSD Sockets
  - Code snippets
  - Performance
- Alternatives to BSD Sockets
  - Network Protocols in User Space

The ISO/OSI reference model

- Communications protocols are divided into independent layers
- Every layer offers a service to the overlying layer

Interplay between OSI layers

- Every layer encapsulates the message into a Protocol Data Unit (PDU)
- PDUs typically consist of a Header and a Data section
- Communication partners exchange PDUs by using the next lower layer
- Receiver unpacks PDUs in reverse order (like a stack)
The TCP/IP model

- The ISO/OSI model is just a theoretical model with almost no implementation
- The most common communications protocols are part of the Internet Protocol Suite (TCP/IP model)
  - Some ISO/OSI layers are merged
  - No strict separation between layers

User Datagram Protocol (UDP)

- UDP is connectionless and unreliable like IP
- Source-Port: The port of the process sending the datagram
- Destination-Port: The port number the datagram should be forwarded to
- Length: The length of the whole PDU in Bytes ($8 < \text{length} < 65535$)
- Checksum: Calculated with the whole PDU and data from the IP header

Transmission Control Protocol (TCP)

- Much more powerful and complex communication service than UDP
- Important application layer protocols based on TCP
  - World Wide Web (HTTP)
  - Email (SMTP)

TCP is reliable:
- Error-free: fragments are retransmitted in case they did not arrive at the destination (timeout)
- preserving order without duplicates
- TCP is connection oriented
  - Connection establishment necessary before data can be sent
  - Connection defined by IP and port number (like UDP) of source and destination
  - Connections are always point-to-point and full-duplex
- It implements flow control and congestion avoidance
- Data is transmitted as an unstructured byte stream
TCP data flow

- A sends frame with SYN and random Sequence number X
- B acknowledges with ACK=X+1 and random Sequence number Y
- A acknowledges the reception
- A sends Z bytes
- B increases the sequence by Z to acknowledge the data reception
- Disconnection works like connection establishment but with FIN instead of SYN

Flow Control and Congestion Avoidance

- Frames are only rarely dropped because of transmission errors (e.g. bit flip)
  - Connections are typically either working without transmission errors or not at all
- Main reason for dropped frames are overloads of the receiver or the network

TCP implements two mechanisms to avoid overloading:

- Flow control: Avoids overloading of the receiver
- Congestion avoidance: Reduces the sending rate in case that fragments are dropped by the network

TCP's Flow Control: Sliding Window

- Each node has a receiving and sending buffer
- In each segment a node specifies how many bytes it can receive
  - Receiver window size: Number of free bytes in the receiving buffer
- If a node has sent as many unacknowledged bytes as the window size is large it will stop sending and wait for the next acknowledgment

TCP's Congestion Avoidance

- Congestion window: Specifies the maximum number of bytes that may be sent without acknowledgment depending on the network capacity
- Max bytes that may be sent = min(sliding win, congestion win)

The congestion avoidance algorithm:

- Initialize the congestion window to typically 2 x MSS (slow start)
- Send until one of the two windows are filled
- If a segment is acknowledged: Increase the congestion window
  - Doubled until threshold reached, then linearly
- If acknowledgment timed out (frame dropped by network):
  - Set threshold to half the current congestion window and go back to slow start
TCP's Congestion Avoidance

Sending Buffer

- When an application sends data chunks to the TCP stack two different approaches can be applied:
  1. Low latency
     - Data chunks sent directly as they are
     - Disadvantage: Many small IP packets will be transmitted (low efficiency)
  2. High throughput
     - Buffer data and send larger segments
     - Higher latency but more efficient

Nagle's Algorithm

- An algorithm to reach the high throughput approach:
  - Send first chunk of data arriving at the TCP stack directly
  - Fill sending buffer with new incoming data without sending
  - If the buffer reaches the MSS: Send a new frame clearing the buffer
  - If all sent segments are acknowledged: Send a new frame clearing the buffer

- Nagle's algorithm is used in almost all TCP implementations
  - Can be deactivated to reduce latency (e.g. for X11 applications)

Switch off Nagle's Algorithm

- This is only rarely necessary!
- Within your program:
  ```c
  int flag = 0;
  setsockopt(socket, IPPROTO_TCP, TCP_NODELAY, (char *)&flag, sizeof(int));
  ```
- System wide:
  ```sh
echo 1 > /proc/sys/net/ipv4/tcp_low_latency
  ```
TCP vs UDP

- TCP: A lot of bookkeeping and additional data transmission for acknowledgments
- UDP: Just sends the data as it is

But...

- TCP: Flow control, congestion avoidance, Nagle’s algorithm

Typical rule of thumb:

- TCP for high throughput, reliability and/or congestion avoidance
- UDP for low latency and broadcasts/multicasts (not possible with TCP)

A Quick RTT Test

This test was performed with hpcbench:

hpcbench.sourceforge.net

UDP vs TCP round trip times

Latency [μs] vs Chunk size [B]

BSD Sockets

- Linux supports TCP/IP as its native network transport
- BSD Sockets is a library with an interface to implement network communications using any TCP/IP layer below the application layer

Important functions

- `socket()` opens a new socket
- `bind()` assigns socket to an address
- `listen()` prepares socket for incoming connections
- `accept()` creates new socket for incoming connection
- `connect()` connects to a remote socket
- `send()` / `write()` sends data
- `recv()` / `read()` receives data
TCP Code Snippet

Simple TCP socket accepting connections and receiving data:

```
socket = socket(AF_INET, SOCK_STREAM, 0);
serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(8080);
serv_addr.sin_addr.s_addr = INADDR_ANY;
bind(socket, (struct sockaddr *) &serv_addr, sizeof(serv_addr));
listen(socket, 5);
connectionSocket = accept(socket, (struct sockaddr *) &cli_addr, &clilen);
recv(connectionSocket, buffer, sizeof(buffer), 0);
```

Complete examples to be found at: http://github.com/JonasKunze

TCP vs UDP: Throughput

- Small frames induce high CPU load → packet loss
- TCP achieves higher throughput

<table>
<thead>
<tr>
<th>Domain</th>
<th>UDP receive rate</th>
<th>UDP send rate</th>
<th>TCP receive rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datarate [Gbps]</td>
<td>12</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>chunk size [B]</td>
<td></td>
<td>1 k</td>
<td>10 k</td>
</tr>
<tr>
<td>CPU load [%]</td>
<td>140</td>
<td>120</td>
<td>100</td>
</tr>
</tbody>
</table>

Interrupt Coalescing

- Technique to reduce interrupt load
- Interrupts are held back until...
  - ...a certain number of frames have been received...
  - ...or a timer times out
- Now the kernel can process several frames at once
  - Higher efficiency with just little increase of latency

```
# print current settings
ethtool -c eth0

# change settings
ethtool -C eth0 rx-usecs 0 # 0 is adaptive mode for many drivers
ethtool -C eth0 rx-frames 12
```
Interrupt Coalescing

- Small values overload the CPU → Packet loss
- High values lead to buffer overflow → Packet loss

NAPI

- An alternative to interrupts is polling:
  - Kernel periodically checks for new data in the NIC buffer
    - High polling frequencies induce high memory loads
    - Low polling frequencies lead to high latencies and packet loss

- NAPI: Linux uses both
  - Interrupts per default
  - Polling in case of high data rates incoming

The kernel still needs to copy incoming data!

Outline

- Recap of the TCP/IP model
  - ISO/OSI and TCP/IP
  - User Datagram Protocol (UDP)
  - Transmission Control Protocol (TCP)
- Network programming with BSD Sockets
  - Code snippets
  - Performance
- Alternatives to BSD Sockets
  - Network Protocols in User Space
    - Example: pf_ring DNA
    - Reliability on top of UDP?
    - Reliability without acknowledgment

Network Protocols in User Space

- Following approach can be implemented in the user space to avoid double copies
  - NIC copies incoming data to a user space buffer (DMA)
  - The user space application polls the buffer
  - The user space application may enable interrupts for low data rates
  - The kernel is only used for the initialization
- 0% CPU used for accessing the data
Example: pf_ring DNA

- Proprietary user space driver by ntop
- Does not implement any protocol
  - You need to implement them: ETH, IP, UDP, TCP, ARP, IGMP...
- Compatible with all 1 GbE and 10 GbE NICs running on PCI-E
- Full line rate (1-10 GbE) with any frame size
- Round trip time below 5 µs
- Hardware filtering (only Intel and Silicom NICs)
  - Very efficient Intrusion prevention systems possible (Snort)
- Other userspace drivers: Netmap, Intel DPDK, OpenOnload

Reliability on top of UDP?

- At CERN experiments most data senders are FPGAs
  - Very fast in parallel jobs
  - Typically fully loaded by algorithms
  - Sometimes there’s no space left for a fully implemented TCP/IP stack
- I’ve seen many groups implementing reliable protocols on top of IP
  - In most cases the result was TCP without flow and congestion control
- Being compatible with TCP/UDP relieves the software developers
  - You don’t need to implement the protocol on the receiver side
  - Instead you can use standard libraries

Reliability without acknowledgment

- Sometimes it’s not even possible to store data until the acknowledgment is received
  - You should use pure UDP in this case
- As soon as datagrams are sent out you have to trust the network
  - Make sure that you don’t overload switches/routers/receiver nodes
  - Check every node whether frames are dropped

Switch/Router:
show interfaces ...

Linux:
cat /proc/net/udp

Summary

- TCP is more than just reliable
  - It implements a maximum efficient data transmission
- BSD sockets provide a nice API for simple network programming
  - For more complex architectures networking libraries are recommended
- Linux’ network sockets are not as efficient as they could be
  - High performance network drivers provide efficient alternatives to BSD sockets but they generate additional work for the developer team
**LECTURE 2**
Is your web API truly RESTful (and does it matter)

**Monday 24 February**

<table>
<thead>
<tr>
<th>15:50-16:50</th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>After a brief introduction to the history of web services, this lecture will cover the basics of REST (Representational State Transfer) and provide you with an understanding of essential terms and constraints. We will have a look at the API design process, and think about what you should consider when designing a scalable web service. Furthermore, we will explore how striving for a resource-oriented client architecture helps to reap the benefits of REST without sacrificing user experience.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15:50-16:50</th>
<th><strong>Audience</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This lecture targets everyone (mainly computer scientists) involved in web development, in particular those responsible for the API design of the back-end. After this lecture, the attendees are expected to have a comprehensive overview of the REST principles and its benefits for modern, scalable web development. Furthermore, the participants will be aware of potential pitfalls and ways to address them.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15:50-16:50</th>
<th><strong>Pre-requisite</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No special prior knowledge is required to follow this lecture. However, a basic understanding of HTTP and web development will prove beneficial.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th><strong>Josef Hammer</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>CERN</strong></td>
</tr>
</tbody>
</table>
Is your web API truly RESTful (and does it matter)?

Josef Hammer
CERN

The Programmable Web

- The “human web” is a great success story
  - Highly scalable
  - Easy to change
  - With only the knowledge of a base URL (e.g. www.cern.ch) you can explore and interact with any web site

- But APIs for machines are more difficult
  - Hard to discover / explore: Machines do not understand the meaning of names
  - Most APIs are difficult to change once deployed

- RESTful architectures provide a solution

Outline

- History
- Introduction to REST
- RESTful API Design
  - URIs
  - HTTP
  - Hypermedia
- Conclusion

Where do we come from?

- COM
  - Component Object Model
- CORBA
  - Common Object Request Broker Architecture
- XML-RPC
  - Extensible Markup Language Remote Procedure Call
- SOAP
  - Simple Object Access Protocol
  - WSDL (Web Services Description Language)
  - Big “service document” → tight coupling, hard to change
Representational State Transfer (REST)

- Term defined in Roy Fielding’s dissertation in 2000 [fielding]
  - A technical description of how the World Wide Web works
- Architectural style, not a protocol like SOAP
  - 6 architectural constraints (“Fielding constraints”)
- Resources + representations
  - “The server sends a representation describing the state of a resource. The client sends a representation describing the state it would like the resource to have. That’s representational state transfer.” [rwa]
- Not limited to HTTP

Fielding Constraints (1) [fielding, rwa]

- Client-server
  - All communication on the web is one-to-one (vs. peer-to-peer w/ multiple sources)
- Stateless
  - When a client is not currently making a request, the server doesn’t know it exists.
- Cacheable
  - A client can save trips over the network by reusing previous responses from a cache.

Fielding Constraints (2) [fielding, rwa]

- Layered system
  - Intermediaries such as proxies can be invisibly inserted between client and server.
- Code on demand (optional)
  - The server can send executable code in addition to data. This code is automatically deployed when the client requests it, and will be automatically redeployed if it changes.
  - E.g. Javascript code in the browser

Fielding Constraints (3) [fielding, rwa]

- The uniform interface
  - Identification of resources
    - Each resource is identified by a stable URI.
  - Manipulation of resources through representations
    - The server describes resource state by sending representations to the client. The client manipulates resource state by sending representations to the server.
  - Self-descriptive messages
    - All the information necessary to understand a request or response message is contained in (or at least linked to from) the message itself.
  - The hypermedia constraint (“HATEOAS”)
    - The server manipulates the client’s state by sending a hypermedia “menu” containing options from which the client is free to choose.
HATEOAS (1)

- Hypermedia As The Engine Of Application State

- “Hypermedia”: Links, basically

- “Clients make state transitions only through actions that are dynamically identified within hypermedia by the server (e.g., by hyperlinks within hypertext). Except for simple fixed entry points to the application, a client does not assume that any particular action is available for any particular resources beyond those described in representations previously received from the server.”

HATEOAS (2)

- “A distributed application makes forward progress by transitioning from one state to another, just like a state machine. The difference from traditional state machines, however, is that the possible states and the transitions between them are not known in advance. Instead, as the application reaches a new state, the next possible transitions are discovered.”

- Clients only need to know the entry point (base URI)

- Clients shall not be required to construct URIs

- Loose coupling → easy to maintain

HATEOAS (3)

REST Maturity Model (RMM) (1)

- by Leonard Richardson

- a.k.a. Richardson Maturity Model

- how “RESTful” is a web API?
REST Maturity Model (RMM) (2)

Level 3: Hypermedia controls
- Level 2 + uses hypermedia for navigation
- `<a href="/slides/2" rel="next">`

Level 2: HTTP methods
- multiple URIs, multiple HTTP methods
- `PUT|DELETE /slides/1`

Level 1: URIs (‘Resources’)
- multiple URIs, single HTTP method
- `POST /slides/1`

Level 0: XML-RPC, SOAP, ...
- single URI, single HTTP method
- `POST /slides`

URI vs URL vs URN

- URI: Uniform Resource Identifier
  - A short string to identify a resource
  - Might have no representation

- URL: Uniform Resource Locator
  - A URI that can be dereferenced (= has a representation)
  - E.g. `http://www.cern.ch`

- URN: Uniform Resource Name
  - no protocol to dereference
  - E.g. `urn:isbn:9781449358063`

URI Design

- “The only thing you can use an identifier for is to refer to an object. When you are not dereferencing, you should not look at the contents of the URI string to gain other information.”
  [Tim Berners-Lee, w3-axioms]

- “That said, REST API designers should create URIs that convey a REST API’s resource model to its potential client developers.”
  [wad]

- “A REST API’s clients must consider URIs to be the only meaningful resource identifiers. Although other backend system identifiers (such as database IDs) may appear in a URI’s path, they are meaningless to client code.”
  [wad]
Is your web API truly RESTful (and does it matter)?

Resource Archetypes
- 4 basic types (+ naming rules)
  - Document
  - Collection
  - Store
  - Controller

HTTP Methods ("Verbs") (1)
- GET: Get a representation of this resource
- POST: Create a new resource
- PUT: Replace the state of (or create) this resource with the given representation
- DELETE: Destroy this resource

HTTP Methods ("Verbs") (2)
- GET: Safe + idempotent: no side effects / state changes allowed!
- POST: Create a new resource
- PUT: Replace the state of (or create) this resource with the given representation
- DELETE: Destroy this resource

HTTP Methods ("Verbs")
- GET: Caching allowed
- PUT: Idempotent (the most generic method)
- DELETE: Idempotent (i.e. repeating the request leads to the same result / state)

Resource Archetypes
- 4 basic types (+ naming rules)
  - Document
  - Single item (noun, sg)
  - Collection of items; server decides on URI (noun, pl)
  - Store
  - Transactions etc. - try to avoid (verbs)

Controller
- Transactions etc.

Document
- Single item (noun, sg)

Collection
- Collection of items; server decides on URI (noun, pl)

Store
- Special kind of collection: item URIs are user-defined

Controller
- Transactions etc.

HTTP Methods ("Verbs") (1)
- GET: Get a representation of this resource
- POST: Create a new resource
- PUT: Replace the state of (or create) this resource with the given representation
- DELETE: Destroy this resource

HTTP Methods ("Verbs") (2)
- GET: Safe + idempotent: no side effects / state changes allowed!
- PUT: Idempotent (the most generic method)
- DELETE: Idempotent (i.e. repeating the request leads to the same result / state)
HTTP Methods („Verbs“) (3)

- **HEAD**
  - Get the headers that would be sent along with a representation of this resource, but not the representation itself. **Safe!**

- **OPTIONS**
  - Discover which HTTP methods this resource responds to

- **CONNECT, TRACE**
  - Used only with HTTP proxies

HTTP Methods („Verbs“) (4)

- **PATCH**
  - Extension defined in RFC 5789
  - Modify *part* of the state of this resource

- **LINK (draft)**
  - Connect some other resource to this one

- **UNLINK (draft)**
  - Destroy the connection between some other resource and this one

CRUD

- **Create, Read, Update, Delete**
  - *everything you need for collections 😊*

  - **Maps perfectly well to HTTP verbs**
    - Create → POST (collection), PUT (store)
    - Read → GET
    - Update → PUT
    - Delete → DELETE

- **Rest Maturity Model Level 2**
  - does not fit everything (limited vocabulary)
  - shared, tightly coupled understanding of resource life

Requests: Good, Bad, or Evil? (1)

- **GET /deleteUser?id=1234**
  - Evil! GET must not modify the resource state!

- **GET /deleteUser/1234**
  - Certainly looks better ;) … nevertheless just as evil!

- **DELETE /users/1234**
  - Method name in URI … bad.

- **POST /users/1234/delete**
  - Why use a controller when there is a standard method? **Bad.**

- **DELETE /users/1234**
  - 😊
Requests: Good, Bad, or Evil? (2)

- GET /users/register
  Assuming “register” means creating a new user:
  Might make sense for a human client (web site).
  In an API: Bad. Retrieve a template with GET /users if necessary.

- POST /users/register
  No need to use a controller for creating a resource ... bad.

- POST /users

- PUT /users
  If you really want to replace/update your entire user database ;)
Conditional Requests (2)

GET /books/27 HTTP/1.1
HTTP/1.1 200 OK
ETag: "a23-45-67c"
{...，“price”: 30, ...}

PUT /books/27 HTTP/1.1
If-Match: "a23-45-67c"
{...，“price”: 29, ...}
HTTP/1.1 412 Precondition Failed

Hypermedia

- “Hypermedia is the general term for things like HTML links and forms: the techniques a server uses to explain to a client what it can do next.”

  E.g. the <a> tag is a simple hypermedia control

- Works well for human clients
  - We simply follow links labelled “Add to Cart”, “Sign In”, ...

- ... but how can we tell machines the semantic meaning of these links?

Link Relations (1)

- Links in many data formats allow the rel attribute
  - Relation between the linked resource and the current one

  E.g. in HTML
  - <link rel="stylesheet" type="text/css" href="/style.css"/>
  - Tells browsers to automatically retrieve /style.css and use it to style the current page

  Communicate the “meaning” of a link to the client
  - Clients can interpret the relation and choose the right link
Is your web API truly RESTful (and does it matter)?

### Link Relations (2)

- HTTP/1.1 200 OK
- Link: <http://.../story/27/part2>;rel="next"

GET /story/27 HTTP/1.1

GET /story/27/part2 HTTP/1.1

If available: follow link with 'next' relation

### Link Relations (3)

- Link relations mean nothing without a formal definition

- RFC 5988 defines 2 types
  - Registered link relations
    - E.g. IANA (Internet Assigned Numbers Authority) manages a registry
    - E.g. self, next, previous
  - Extension relations
    - Like URLs – you are allowed to define anything within your domain
    - E.g. http://josefhammer.com/toc

### Evolvable APIs (1)

- Decoupling the client from the server
  - Use link relations instead of hard-coded / constructed links
  - Choose from the set of provided links only
  - ... allows APIs to evolve
  - URIs can be changed → only the relation is hard-coded
  - Features can be added → old versions of the client will ignore unknown links
  - Features can be removed → clients gracefully ignore missing links

### Evolvable APIs (2)

- POST /bugs HTTP/1.1
  - \{ "description": "..." \}

HTTP/1.1 201 CREATED
Location: /bugs/42

\{ "bugID": 42,
  "links": [
    \{ "rel": "self",
      "href": "/bugs/42" \},
    \{ "rel": "reject",
      "href": "/bugs/42/rejection" \},
    \{ "rel": "fix",
      "href": "/bugs/42/solution" \}
  ]
\}

No hard-coded links in the client
Evolvable APIs (3)

POST /bugs HTTP/1.1
{ "description": "" }

HTTP/1.1 201 CREATED
Location: /bugs/43
{ "bugID": 43,
"links": [ { "rel": "self",
"href": "/bugs/43" },
{ "rel": "comment",
"href": "/bugs/43/comments" } ]
}

Domain specific data formats

- Try to exploit existing domain specific data formats
  - Atom, AtomPub
  - OData
  - Collection+JSON
  - OpenSearch
  - ...Microformats
  - HTML Microdata

- Client tools may exist
- Developers more likely to be familiar with the terms

Evolvable APIs (4)

POST /bugs HTTP/1.1
{ "description": "" }

HTTP/1.1 201 CREATED
Location: /bugs/44
{ "bugID": 44,
"links": [ { "rel": "self",
"href": "/bugs/44" },
{ "rel": "comment",
"href": "/bugs/44/comments" },
{ "rel": "attach",
"href": "/bugs/44/attachments" } ]
}

Microformats

- E.g. the hcard microformat

  `<div class="vcard">
  <span class="n">
  <span class="given-name">Josef</span>
  <span class="family-name">Hammer</span>
  </span>
  <span class="given-name">Josef</span>
  <span class="family-name">Hammer</span>
  </div>`

- Well-defined and understood terms
- Easy to embed in HTML
- `microformats.org` provides a collection of schemata
Microdata

- A refinement of the microformat concept for HTML 5

- 5 new attributes for any HTML tag
  - `itemscope` Starts a new scope (boolean)
  - `itemprop` Like `class` in HTML
  - `itemtype` Where to find the type definition
  - `itemid` Global identifier (valid URL)
  - `itemref` List of itemIDs

- `schema.org` provides a collection of schemata

Conclusion

- Use hypermedia + well-known terms & concepts
  - Allows machines to discover and adapt to changing web APIs
  - (Partial) solutions may be available already
  - Less (learning) effort for developers

- “REST is not the answer to all questions. […] But in order to explore these boundaries properly, it’s vital to have a proper understanding of what REST is about. Without that, you run the risk of trying pseudo-REST and drawing the wrong conclusions.”
  - [Martin Fowler, rip](rip)

References

- fowler-rmm: http://martin Fowler.com/articles/RichardsonMaturityModel.html
- hcard: http://microformats.org/wiki/hcard
- rad: REST API Design Rulebook. Mark Maier; O’Reilly, October 2011
- rip: REST in Practice. Jim Webber, Savas Parastatidis, Ian Robinson; O’Reilly, September 2010
- rwa: RESTful Web APIs. Leonard Richardson, Mike Amundsen, Sam Ruby; O’Reilly, September 2013
- rwc: RESTful Web Services Cookbook. Subbu Allamaraju; O’Reilly, March 2010
- uri: http://en.wikipedia.org/wiki/URI_Euler_Diagram_no_lone_URIs.svg
- w3-axioms: http://www.w3.org/DesignIssues/Axioms.html
- waa: Designing Evolvable Web APIs with ASP.NET. Glenn Block, Pablo Cibrano, Pedro Felix, Howard Dierking, Darrel Miller; O’Reilly, March 2014 (est.; early release March 2013)
### LECTURE 3
**Building highly distributed systems within 5 minutes**

**Monday 24 February**

<table>
<thead>
<tr>
<th>16:50-17:50</th>
<th>Description</th>
<th>Jonas Kunze</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Highly distributed systems are typically very complex. Traditionally, it took a long time to design the dataflow and it may have taken even more time to implement the necessary communication interfaces. But using modern libraries to outsource the communication significantly reduces development and implementation time. During this lecture several communication patterns will be discussed and a selection of libraries for different application areas will be introduced: <strong>Boost.Asio</strong> is a C++ library that provides you a consistent way to develop asynchronous communication and therefore makes it easy to develop a highly parallel program. <strong>ØMQ</strong> is a library for many different programming languages. It provides the distribution of messages with several patterns and therefore clearly facilitates the development of distributed systems. <strong>Apache Thrift</strong> is a framework enabling Remote Procedure Calls (RPCs) between many different languages. It generates source code for the server and client based on a given interface description file.</td>
<td><strong>Johannes-Gutenberg Universität Mainz - DE</strong></td>
</tr>
</tbody>
</table>

| **Audience** | This lecture is addressed to physicists and engineers that need to implement distributed systems. During this lecture the attendees will get an overview of different modern approaches to implement distributed systems. |  |
| **Pre-requisite** | A basic programming knowledge is recommended to follow this lecture. Additionally a basic understanding of the main Internet Protocols is advisable. This knowledge can be obtained during the first lecture. |  |
Network Programming
Lecture 2
Building Highly Distributed Systems
Within 5 Minutes
Jonas Kunze
University of Mainz – NA62

Outline
- Motivation
- Boost.Asio
- Message Passing
- ØMQ
- Apache Thrift

TCP in C code
- The BSD socket API is minimalistic
  - No intrinsic multithreading support
  - Handling multiple connections typically via fork()
  - No data management (messaging)
  - Configuration a bit awkward
- There is no exception handling or OOP in C
- There is no C++ socket API in the std library
  - std::socket will never come
Boost.Asio

- Boost.Asio is a C++ library for low-level I/O programming with a consistent **asynchronous** model including a BSD socket interface

### BSD Socket API (Linux) vs. Boost.Asio

<table>
<thead>
<tr>
<th>BSD Socket API (Linux)</th>
<th>Equivalents in Boost.Asio</th>
</tr>
</thead>
<tbody>
<tr>
<td>socket descriptor – int</td>
<td>For TCP: ip::tcp::socket For UDP: ip::udp::socket</td>
</tr>
<tr>
<td>sockaddr_in, sockaddr_in6</td>
<td>For TCP: ip::tcp::endpoint For UDP: ip::udp::endpoint</td>
</tr>
<tr>
<td>accept()</td>
<td>For TCP: ip::tcp::acceptor::accept()</td>
</tr>
<tr>
<td>bind()</td>
<td>For TCP: ip::tcp::socket::bind() For UDP: ip::udp::socket::bind()</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Boost.Asio: Asynchronous operations

- I/O objects implement non-blocking/asynchronous operations
  - E.g. `boost::asio::ip::tcp::socket::async_connect`
- Completion handler function passed to `async_` functions
- `io_service.run()` calls the completion handler as soon as results of `async_` functions are available

```cpp
callIOObject(YourProgram, I/O Object, Operating System, Completion Handler)
```

### Code Example

```cpp
void MyClass::handle_connect(const boost::system::error_code& error) {
  if (!error) { doSomething(); }
}

... 

socket.async_connect(socket, *resolver.resolve({hostname, portNum}),
boost::bind(&MyClass::handle_connect, this,
boost::asio::placeholders::error));

workWhileConnecting();

// Runs handle_connect as soon as the connection is established
```
Concurrency without threads
Handling multiple TCP connections

- One `io_service` can handle several I/O objects and async_operations
- `io_service::run()` will block until all requests have been handled

```cpp
sock1.async_read_some(readBuffer, [](boost::system::error_code error, std::size_t){
  if (!error) {std::cout << "Socket 1 received something" << std::endl;} });

sock2.async_read_some(readBuffer, [](boost::system::error_code error, std::size_t){
  if (!error) {std::cout << "Socket 2 received something" << std::endl;} });

io_service.run();
```

```
cout << "Both sockets received something" << endl;
```
Message passing via TCP

- TCP only offers a continuous data stream
  - Although data is typically sent to sockets in chunks, the receiver may see different chunks (scaling window)
  - The application layer program has to split the stream into messages
- There are three possible approaches to indicate messages in the stream:
  - Protocol defines the message length implicitly
  - The message length is explicitly specified in a message header
  - Line-Based: Messages in the stream are separated by delimiters

- Line-Based approach easily implemented with Boost.Asio:
  ```cpp
  boost::asio::read_until(socket, msgBuffer, "\r\n");
  ```
- Other approaches are much more efficient
  - But also hard work to implement

No need to reinvent the wheel!

ØMQ

- What ØMQ says about their socket library (http://zguide.zeromq.org/page:all):
  We took a normal TCP socket, injected it with a mix of radioactive isotopes stolen from a secret Soviet atomic research project, bombarded it with 1950-era cosmic rays (...) It’s sockets on steroids.
ØMQ

- ØMQ offers a uniform API (ØMQ sockets) to transport messages over different channels:
  - TCP, multicast, IPC (process to process), inproc (thread to thread)
- Cross Platform (Linux, Windows, Mac, etc...)
- Implementations in many(!!!) different languages:
  - C/C++, Java, Python, Ruby, PHP, Perl, Node.js, C#, Clojure, CL, Delphi, Erlang, F#, Go, Haskell, Haxe, Lua, Objective-C, Q, Racket, Scala...
- OpenSource

ØMQ – Messaging Patterns

- ØMQ sockets express several messaging patterns
  - REQ and REP
  - PUB and SUB
  - PUSH and PULL
  - REQ and ROUTER
  - DEALER and REP
  - DEALER and ROUTER
  - DEALER and DEALER
  - ROUTER and ROUTER
  - PAIR and PAIR

ØMQ – Simple REQ Client

```c
int main() {
    zmq::context_t context(1);
    zmq::socket_t socket(context, ZMQ_REQ);
    socket.connect("tcp://REPServerHostName:5555");
    zmq::message_t request(6);
    memcpy((void *) request.data(), "Hello", 5);
    socket.send(request);
    zmq::message_t reply;
    socket.recv(&reply);
    return 0;
}
```
Network Programming

Lecture 2

Building Highly Distributed Systems Within 5 Minutes

ØMQ – Simple REP Server

```c
int main() {
    zmq::context_t context(1); // Similar to io_service
    zmq::socket_t socket(context, ZMQ_REP);
    socket.bind("tcp://*:5555");

    while (true) {
        zmq::message_t request;
        socket.recv(&request);

        zmq::message_t reply(5); memcpy((void *) reply.data(), "World", 5);
        socket.send(reply);
    }
    return 0;
}
```

ØMQ – REQ-REP Notes

- The REQ-REP socket pair is in lockstep
  - Server and client have to call send and recv alternately
  - Server automatically sends to the node it got the last message (recv) from
    - All the connection handling is done by ØMQ

- The connection can be established from both sides (true for all patterns)

ØMQ – PUB-SUB

- Server PUBlishes data to all connected clients
- Clients SUBscribe to the data by connecting to the server
- Subscription to messages by data prefix (filter)
- If no client is connected the data will be lost

ØMQ – Pipeline

- Ventilator: Produces task that can be processed in parallel
- These tasks are then PUSHed evenly to the connected Workers
- After processing the tasks the Workers push the results to a Sink
- Basic load balancing
int main() {
    zmq::context_t context(1);
    zmq::socket_t ventilatorSocket(context, ZMQ_PULL);
    ventilatorSocket.connect("tcp://ventilator:5557");
    zmq::socket_t sinkSocket(context, ZMQ_PUSH);
    sinkSocket.connect("tcp://sink:5558");

    while (1) {
        zmq::message_t task;
        ventilatorSocket.recv(&task); // PULL
        zmq::message_t result = doSomeWork(task);
        sinkSocket.send(result); // PUSH
    }
}
Now you only have to change one line in the ventilator:
socket.bind("tcp://*:5559");  \rightarrow  socket.connect("tcp://broker:5559");

And connect the worker to the broker instead of the ventilators
ventilator.connect("tcp://ventilator1:5557");
ventilator.connect("tcp://ventilator2:5557");  \ldots

Turns to:
ventilator.connect("tcp://broker:5557");

And again you can start ventilators, workers, broker and sink in whatever order you like:

Messages are queued as close to the receiver as possible

So far we used:
socket.bind("tcp://*:5555");

to run the same programs locally one should use:
socket.bind("ipc://tmp/helloWorld");  // For processes
socket.bind("inproc://helloWorld");  // For threads

Start developing your software with many modules communicating with IPC
Then outsource heavy loaded services to external boxes just by changing
inproc/ipc://...  \rightarrow  tcp://...

ØMQ sockets are not thread safe!
But they are extremely lightweight
  - Create one (or more) sockets per thread
  - Use these ØMQ sockets to exchange messages between the threads
  - Use a proxy to distribute work among the threads

void workerThread(zmq::context_t& context) {
  zmq::socket_t ventilatorProxy(context, ZMQ_PULL);
  ventilatorProxy.connect("inproc://workers");

  zmq::socket_t sink(context, ZMQ_PUSH);
sink.connect("tcp://sink:5558");

  while (1) {
    zmq::message_t task;
    ventilatorProxy.recv(&task);

    zmq::message_t result = doSomeWork(task);
sink.send(result);
  }
}
`ØMQ – Multithreaded Worker`

```c++
int main() {
    zmq::context_t context(1);
    zmq::socket_t ventilatorProxy(context, ZMQ_PULL);
    ventilatorProxy.connect("tcp://localhost:5560");
    zmq::socket_t workers(context, ZMQ_PUSH);
    workers.bind("inproc:///workers");
    std::vector<std::thread> threadPool;
    for (std::size_t i = 0; i < std::thread::hardware_concurrency(); ++i) {
        threadPool.push_back(std::thread(
            [&] () {
            workerThread(context); // will connect with inproc:///workers
        }));
    }    
    zmq::proxy(ventilatorProxy, workers, NULL);
}
```

`ØMQ – Notes`

- With ØMQ messages still need to be translated to procedure executions
- Object serialization has to be implemented on top of ØMQ
- There’s much more functionality in ØMQ!
- Read the great guide: [http://zguide.zeromq.org](http://zguide.zeromq.org)
- The examples in this lecture are based on the examples from the zguide

`Outline`

- Motivation
- Boost.Asio
- Message Passing
- ØMQ
- Apache Thrift

`Apache Thrift`

- Remote Procedure Calls (RPCs):
  Executing subroutines (functions, methods) on a program running remotely
- Thrift is a scalable cross-language RPC framework developed by Facebook
  - It implements the missing object serialization
  - It does not offer
- It’s an open source project in the Apache Software Foundation
Apache Thrift

- The developer defines services in an Interface Definition Language (IDL) file
- Thrift generates code (Interfaces) to be used to call these services remotely
  - E.g. calling a Java Method from a PHP script running on a remote host

Thrift – Interface Definition

- Interface Definition Language (.thrift) files
  - Define namespace, data structures, types, methods, services
  - Similar to C syntax
  - Basic types are bool, byte, i16/32/64, double, string, map<t1,t2>, list<t1>, set<t1>

  namespace cpp ch.cern.icsc14
  struct Work {
    1: i32 num1,
    2: i32 num2,
    3: Operation op
  }
  enum Operation {
    ADD = 1,
    SUBTRACT = 2,
    MULTIPLY = 3,
    DIVIDE = 4
  }
  service Calculator {
    i32 calculate(1:Work w)
  }

Thrift – Compiling Thrift Files

- Thrift compiles the IDL files to server (and client) source code
- It generates thousands of lines of code with placeholders

  Calculator_server.skeleton.cpp:
  using namespace ::ch::cern::icsc14;
  class CalculatorHandler : virtual public CalculatorIf {
    public:
    CalculatorHandler() {
      // Your initialization goes here
    }
    i32_t calculate(const Work& w) {
      // Your implementation goes here
    }
  };

Thrift – Documentation

- There is only very little documentation online
- Useful links:
  - http://wiki.apache.org/thrift/ThriftUsage
  - http://thrift-tutorial.readthedocs.org/
  - http://www.slideshare.net/dvirsky/introduction-to-thrift

Good Luck!
Summary

- There is no native C++ library for network programming
- There are many different libraries for different purposes
  - Boost.Asio for easy asynchronous and multithreaded socket programming
  - ØMQ additionally provides message passing and helpful patterns
  - Apache Thrift provides an efficient RPC framework
- All these libraries are cross-platform capable
- ØMQ and Thrift provide interfaces for many languages

Visit https://github.com/JonasKunze for code snippets and these slides.
LECTURE 4
From Quark to Jet: A Beautiful Journey
Beauty physics, tracking and large-scale distributed computing in HEP

Tuesday 25 February

<table>
<thead>
<tr>
<th>Time</th>
<th>Description</th>
<th>Tyler Mc Millan Dorland</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00-10:00</td>
<td>The beauty, or bottom, quark is an extremely powerful object in high energy physics. Distinctive characteristics of the decay of the quark have motivated the design high-energy physics detectors in the quest to reconstruct the quark. Each component of the detector is used as a tool to identify these characteristics and present unique challenges that are solved with a combination of engineering, physics, and computing. These two lectures will explore the tools used and the different physics and computing environments they are used in, and the important interplay between the two environments and limitations they set on each will be considered. When all components are reassembled, the quark is described as a jet. This lecture will briefly introduce beauty physics and it's role in HEP. Attention will be paid to the distinctive features of the decay of the beauty quark, such as secondary vertices, leptonic decay, and hadronization features that require all subsystems of a detector to be used in it's reconstruction. Then an introduction to a track reconstruction algorithm with particular attention paid to the performance and precision of different seeding and/or iterative methods. The implementation of the algorithms into an experiment wide software infrastructure, and the requirements of large-scale computing, will be explored. The performance of these on large scale computing infrastructures will be presented. Finally, the expected impact of future LHC running and the challenges this presents to current algorithms and performance, along with the concepts used to solve these problems are presented.</td>
<td>DESY - Hamburg</td>
</tr>
</tbody>
</table>

**Audience**
The target audience of this lecture is broad between physicists and computer scientists, but perhaps more focused on computer scientists with only some knowledge of object reconstruction.

**Pre-requisite**
As prerequisite this lecture will not require specific knowledge, but will include some high-level explanations for those with extensive knowledge of physics detectors or distributed computing.
From Quark to Jet: A Beautiful Journey

Lecture 1

Beauty Physics, Tracking, and Distributed Computing

Tyler Dorland
Deutsches Elektronen-Synchrotron (DESY)

Explaining the Title: An outline

- **Theory**
  - Quarks: Mathematical Representation
    - Matrices, operators, etc.
  - Huge numbers of complex equations

- **Hadronization**
  - Particles: intermediate and final state objects
  - Entirely Simulated, particles are subjected to decay conditions

- **Reconstruction**
  - Jets: Energy deposited in detector, algorithms used to recreate particles
  - Detector simulation, Algorithmic reconstruction
  - Distributed Computing, Laptops, Local Servers

Inverted CERN School of Computing, 24-25 February 2014
iCSC2014, Tyler Dorland, DESY
Explaining the Title: An outline

- Theory
- Hadronization
- Reconstruction

Quarks/Mathematical Representation
Matrices, operators, etc.

Particles: intermediate and final state objects

Jets: Energy deposited in detector, algorithms used to recreate particles

Huge numbers of complex equations

Entirely Simulated, particles are subjected to
Detector simulation, Algorithmic reconstruction

Distributed Computing, Laptops, Local Servers

Ask Questions here

- Use theory to make predictions for observables of particles
- Design detectors to detect these observables
- Algorithms to remake the objects

Beauty Physics - Theory

- Beauty quark discovered in 1977 at Fermilab
- First third generation quark
- Beauty (and charm) quarks have a lifetime that allows for decay lengths of a few millimeters
  - Top is too short, up/down/charm is too long

- b-jets are extremely powerful background reducers
  - Many important signals have b-quarks
  - Huge order of magnitude reduction from identifying b-quarks
  - Very important tool

Beauty Physics - Theory

pp->anything

10 x pp->X-> beauties

> 10 orders of magnitude

2 orders of magnitude
Hadronization

- Most calculations are confined to simple elements
- What we actually measure is much more complicated

Hardonization

- We know that as quarks get further away from each other they make pairs with other quarks
  - These are called hadrons
- Hadronization depends on many experimentally adjusted factors
- Most importantly we can begin to look at event topology

Beauty Physics – Particle Level

- If a b-quark is paired with an s-quark the resulting meson, B_s, has a long lifetime, and some very interesting decay signatures
- We use these particular decay signatures to determine what experimental signature we want to see

Experimental signature

- Now we have a distinct signature to search for
  - A secondary vertex
  - Jet
  - Displaced track
  - Lepton
- Rare, but not unique
  - We will use different techniques to classify
  - Essentially a probability the jet came from a b-quark
Use theory to make predictions for observables of particles
Design detectors to detect these observables
Reconstruction algorithms to remake the objects

An nth degree polynomial will exactly fit (n+1) points
Therefore, any three points can be fit with a circle
Fits generally classified by distances of points to fitted curve (chi-squared)
For nth degree polynomial, n+2 ... n+m points are degrees of freedom
Track Seeding and Reconstruction

- Inside the collision region we will have many hits we can associate with a primary vertex.

Track Seeding and Reconstruction

- Choose an initial set of layers that we name the "seeding layers" that provide an initial estimate of track parameter.
- Then collect all possible hits associated with different seeds.
- Using techniques to estimate the goodness of the fit we can then estimate the final track parameters.
Fake Removal

- Choose an initial set of layers that we name the “seeding layers” that provide an initial estimate of track parameters.
- Then collect all possible hits associated with different seeds.
- Using techniques to estimate the goodness of the fit we can then estimate the final track parameters.
- And remove hits not associated with good tracks.

Iterative Tracking

- With iterative tracking certain quality tracks can be chosen and then removed from further inspection.
- Then use the remaining hits to create the remaining tracks.
- After many iterations we end with the final set of tracks.

Real data examples
Looking Towards 2015

Current algorithms were developed considering the run conditions for 2011-2012 where there was an average of 20 interactions per bunch crossing.

For 2015, there could be over 40 interactions on average.

With no changes, the computing power needed could be 6 times what is currently used.

Tracking

- Charged particles make curves in magnetic fields

CMS Computing Network

CMS Current Event Model

- Global configurations are loaded into memory
  - Then configurations specific to the specific time of running
- Events then processed serially
- The most time intensive part of event reprocessing is tracking
Amdahl’s Law

- Amdahl’s Law is the upper limit on the speedup gained by a number of processors

\[ S(N) = \frac{1}{(1 - P) + \frac{P}{N}} \]

CMS Threaded Design

- Events are not seen globally
- Multiple events are run concurrently
  - Less backup from very complicated events
- Streams still process serially

Concurrent Processing Inside an Event

- Current event Processing

  ![Diagram of concurrent processing]

- Threading inside of a module

  ![Diagram of threading inside a module]
Conclusions

- Beauty physics is a very diverse and large part of high energy physics
- B-Hadrons have distinguishing traits we can use to make b-jets very powerful tools for background reduction
- To make use of this, we must use information from many parts of the detector, which all require their own reconstruction algorithms and different levels of computing resources
- By restructuring the event processing structure to accommodate threaded applications, we can meet the demands required for tracking in the future

Performance Results

- Single threaded runs out of memory at 3000 simultaneous events
- Definite improvement through multithreading
**LECTURE 5**  
Read-Out Electronics: where data come from

<table>
<thead>
<tr>
<th>Tuesday 25 February</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:00-11:00</td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>This lecture gives a general overview on the evolution of detectors used in HEP and on the necessity of electronics in modern set-ups. The second part of the lecture is focused on the Read-Out Electronics (ROE) as a transmission line: information are transported from the detector to the Data Storage; at high frequency it is important to take into account particular characteristics (such as signal propagation and integrity, interaction between signals and electromagnetic interferences) of the signal even if it is a digital one.</td>
</tr>
<tr>
<td><strong>Audience</strong></td>
</tr>
<tr>
<td>This lecture targets everyone interested in the basic concepts of transmission lines. After this lecture the attendees are expected to have acquired a basic knowledge of the important concepts, properties and limitations of transmission lines (bandwidth, impedance, crosstalk, etc...) and of the standard read-out electronics used in modern HEP experiments.</td>
</tr>
<tr>
<td><strong>Pre-requisite</strong></td>
</tr>
<tr>
<td>This lecture can be followed by anyone having interest in the subject and basic knowledge in electronics and detectors. Nevertheless the main concepts explained in this lecture will be understandable by everyone.</td>
</tr>
<tr>
<td><strong>Francesco Messi</strong></td>
</tr>
</tbody>
</table>
| Rheinische Friedrich-Wilhelms University  
Bonn - DE |
Read-Out Electronics: where data come from

Lecture 1

Francesco Messi
Rheinische Friedrich-Wilhelms-Universität Bonn – DE

Outlook

- Why we build detectors
  - We want to study what we cannot simply see with our eyes
  - For that we "ask" the detectors to record a scene
  - So that we can analyze it

As humans, we are curious: we want to know the place we live in, we want to understand what the matter is made of!!!

For that, we build machines (experiments and detectors) to investigate the matter

...it is important to use correctly the tools we have!!!


An event in the ATLAS detector
Research in Physics

a very general overview of the evolution of the detectors used in experimental set-ups

Discovery of “elementary” particles…

<table>
<thead>
<tr>
<th>Year</th>
<th>Particle</th>
<th>Instrument</th>
</tr>
</thead>
<tbody>
<tr>
<td>1947</td>
<td>π⁺ and π⁻</td>
<td>Nuclear Emulsion</td>
</tr>
<tr>
<td></td>
<td>π⁺</td>
<td>Cloud Chamber</td>
</tr>
<tr>
<td>1947</td>
<td>K⁺ and K⁻</td>
<td>Nuclear Emulsion</td>
</tr>
<tr>
<td></td>
<td>K⁺</td>
<td>Cloud Chamber</td>
</tr>
<tr>
<td>1953</td>
<td>Σ⁺</td>
<td>Nuclear Emulsion</td>
</tr>
<tr>
<td></td>
<td>Σ⁻</td>
<td>Cloud Chamber</td>
</tr>
<tr>
<td></td>
<td>Ξ⁻</td>
<td>Bubble Chamber</td>
</tr>
<tr>
<td></td>
<td>Ξ⁺</td>
<td>Cloud Chamber</td>
</tr>
<tr>
<td></td>
<td>Anti-π⁻</td>
<td>Nuclear Emulsion</td>
</tr>
</tbody>
</table>

Research in Physics

A look at yesterday…

- **Nuclear emulsion:**
  - Photographic emulsion were sent to the high atmosphere
  - Interaction with the cosmic rays would leave a track
  - Studying the tracks it is possible to identify different particles and decays:
    - The grain density is proportional to the energy loss by ionization (Bethe-Bloch) direction and (≈)energy
    - A big deviation from a trajectory interaction and new particles
    - A neutral particle does not leave a signature

  e.g. used today in the OPERA experiment (Oscillation Project with Emulsion-Racking Apparatus)

Nuclear physics research was based on observation of tracks from charged particles.
Research in Physics

A look at today…

- Experimental set-up made of a large number of detectors
  - Tracker: detecting the passage of a charge particle ideally without perturbation of its trajectory
    - (e.g. silicon strips or pixels)
  - Calorimeter: measuring the energy released by a particle in the interaction with the detectors
    - (e.g. inorganic scintillators)
  - Time of Flight: suited to measure the time of flight of the particle
    - Always at least in couple: one start and one stop time needed
    - etc…
  - Each detector can acquire data with a rate of MHz or more…
  - Usage of fast electronics is needed to read the data.

Research in Physics

A brief summary…

- In the last century there was a big evolution of detectors
- Nuclear physics research was mainly based on the observation of tracks from charged particles
- Detectors were “purely analog” devices
- Each detector was built to perform the full experiment
- The search for the constituent of the matter requires:
  - high energies,
  - very short times,
  - “rare channels”
The electronics

The Read-Out Electronics, used to communicate with the detectors, is the tool used to transport information from the analog world (spoken by detectors) to the digital world (spoken by the Data Storage System).

The Read-Out Electronics

From a little, fast electrical signal to a bit-stream of data

Front-End Electronics: from the analog to the digital world
- Comparators
- Analog to Digital Converters (ADC)
- Time to Digital Converters (TDC)

Signal processing: the FPGA
- Where they are used...
- For what...
- Why...

In two major steps:

In lecture 2
From the analog to the digital world…

1) Comparator
- Digital “ONE” if the signal is above a fixed threshold
- Amplitude can vary to any value
- Shape can vary
- Applying a threshold we decide (discriminate) what is noise and what is a good signal

2) Analog to Digital Converter: ADC
- Digital word proportional to the charge of the analog signal
- Bandwidth gives information about how fast is the device
- Sampling Rate gives information about how accurate is the digitalization of the signal seen by the device

3) Time to Digital Converter: TDC
- Digital word proportional to the time of the signal
From the analog to the digital world…

Information in binary logic…

- Digital signal → two states: present or absent, “1” or “0”

<table>
<thead>
<tr>
<th>Family</th>
<th>One (V)</th>
<th>Zero (V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTL</td>
<td>2 to 5</td>
<td>0 to 0.8</td>
</tr>
<tr>
<td>NIM</td>
<td>3 to 12</td>
<td>-2 to 1.5</td>
</tr>
<tr>
<td>ECL</td>
<td>-0.81 to -1.13</td>
<td>-1.48 to -1.96</td>
</tr>
<tr>
<td>LVDS</td>
<td>p 1.27 to 2.40</td>
<td>0.92 to 1.12</td>
</tr>
<tr>
<td></td>
<td>n 0.92 to 1.12</td>
<td>1.27 to 2.4</td>
</tr>
</tbody>
</table>

…and many more…

Characteristics of an electronics chain

What do we want from our electronics chain?

- to receive the correct information → Reliability
- to receive the full information transmitted → Efficiency
- to understand the message → No distortion
- we want large bandwidth

The electronics

A brief summary…

- Electronics is needed to read out the information from detectors
- Several stages of different electronics are needed
- Final goal is to convert information from a little, fast analog signal in a bit-stream of data
- The electronics chain must:
  - Preprocess the signals w/o distortion
  - Be reliable
  - Be efficient
  - Be fast
  - Be calibrated to the experiment!!!

An example…

Particle identification can be made in many ways and with different set-ups; in this example protons and positive pions are identified by the measure of the Time of Flight in correlation with their momentum.

The electronics used for the measurement of the time can be crucial for the kind of physics one wants to investigate…
An example: PID using ToF

**Particle Identification**

- Imagine you need to identify Protons from Pions ($\pi^+$)
- We know from particle physics that, for a fixed momentum, the velocity of the two particles are different
- \( \Rightarrow \) we can identify the particles measuring the Time of Flight

---

**Set-up principle**

- Tracker detectors + fixed magnet = momentum spectrometer
- Time\(_{\text{Stop}}\) - Time\(_{\text{Start}}\) = Time of Flight (ToF)

---

**Separation Power**

\[ SP = \frac{\langle x \rangle_1 - \langle x \rangle_2}{\sigma_1^2/2 + \sigma_2^2/2} \]
An example: PID using ToF

Set-up principle

- It is a simple measurement, but involves a lot of electronics

Detector 1

Amp.

Discr.

Time of Flight

Detector 2

Amp.

Discr.

start

TDC

stop

- How much does the time resolution of the stop detector influence the measure? Let’s simulate different resolutions...

An example: PID using ToF

1) Ideal electronics: no spread of time due to the TDC

TrackTime [ns]

0.5 1 1.5 2 2.5 3

Particle momentum [GeV/c]

0.5 1 1.5 2 2.5 3

TrackTime [ns]

-10 0 10 20 30 40 50

Number of Entries

0 10 20 30 40 50 60

Projection Y of bin x = [150, 159]

SP = 14.3

SP = 4.7

An example: PID using ToF

2) A “good” electronics: ~480 ps time resolution

TrackTime [ns]

-2 0 2 4 6

Particle momentum [GeV/c]

0.5 1 1.5 2 2.5 3

TrackTime [ns]

-2 0 2 4 6

Number of Entries

0 10 20 30 40 50

Projection Y of bin x = [290, 299]

SP = 8.9

SP = ???

An example: PID using ToF

3) A “bad” electronics: ~1.9 ns time resolution

TrackTime [ns]

-3 -2 -1 0 1 2 3 4 5 6

Particle momentum [GeV/c]

0.5 1 1.5 2 2.5 3

TrackTime [ns]

-3 -2 -1 0 1 2 3 4 5 6

Number of Entries

0 10 20 30 40 50 60

Projection Y of bin x = [200, 209]

SP = 2.9

SP = ???
An example: PID using ToF

Conclusions...

- We want to identify Protons and $\pi^+$ using a Time of Flight
- From the plot “ToF vs Momentum” we can distinguish the two particles only if we can separate the two time distributions
- The time resolution of the electronics chain affects which kind of physics you can investigate
  - Different technologies allow for different time resolutions…
  - …but budgets are limited
- You must choose your electronics taking into account your specific physics program!!!
From Quark to Jet: A Beautiful Journey
Making a jet, classifying a jet, and personal scale computing in HEP

Tuesday 25 February

<table>
<thead>
<tr>
<th>11:30-12:30</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This lecture will begin by presenting the detecting systems needed for proper jet reconstruction. Then the jet reconstruction algorithms in detail are explained. A brief explanation of combination algorithms follows, completed by an examination of the datastructures used to store them. This will be followed by a brief description of a multivariate technique commonly used for the classification of these objects and the software package (TMVA) used to implement them. Particular attention is paid to which parts of these algorithms can be completed on university or personal level computing that is the common interface for analysts.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Audience</th>
</tr>
</thead>
<tbody>
<tr>
<td>The target audience of this lecture is broad between physicists and computer scientists, but perhaps more focused on physicists with some knowledge of different computing environments. The benefits of following this lecture is the understanding of jet reconstruction algorithms, understanding of personal/group level computing in HEP, understanding of a single multivariate analysis technique (most likely decision trees), understanding of data structures used in HEP (ROOT based), and new approaches in common software at an experiment wide level.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pre-requisite</th>
</tr>
</thead>
<tbody>
<tr>
<td>As prerequisite some knowledge of a particle will be helpful but not required. This lecture will be reasonably independent of the first lecture; however the physics motivation for b-quarks will not be revisited extensively.</td>
</tr>
</tbody>
</table>

Tyler Mc Millan Dorland
DESY - Hamburg
From Quark to Jet: A Beautiful Journey
Lecture 2

Jet Clustering, Classification, and Personal Computing

Tyler Dorland
Deutsches Elektronen-Synchrotron (DESY)

Explaining the Title: An outline

- Theory
  - Quarks: Mathematical Representation
    Matrices, operators, etc.
  - Quarks: Large numbers of complex equations
  - Entirely simulated, particles are subjected to decay conditions

- Hadronization
  - Particles: intermediate and final state objects
  - Hadronization schemes
  - Experimental uncertainties

- Reconstruction
  - Jets: Energy deposited in detector, algorithms used to recreate particles
  - Detector simulation, Algorithmic reconstruction
  - Distributed Computing, Laptons, workgroup servers
Explaining the Title: An outline

- **Theory**
  - Quarks: Mathematical Representation
    - Matrices, operators, etc.
  - Huge numbers of complex equations

- **Hadronization**
  - Particles: intermediate and final state objects
  - Entirely Simulated, particles are subjected to
  - Detector simulation, Algorithmic reconstruction

- **Reconstruction**
  - Jets: Energy deposited in detector, algorithms used to recreate particles
  - Distributed Computing, Laptons, workgroup servers

---

**Ask Questions here**

- Use theory to make predictions for observables of particles
- Design detectors to detect these observables
- Reconstruction algorithms to remake the objects

---

**Calorimeters**

- Calorimeters are designed to capture the energy of particles
- Two different types are in use at LHC
  - Homogeneous – Capture all of the energy of the incident particle
  - Sampling – capture a portion the incident energy and make a correction

**Sampling Calorimeter**

- Sampling calorimeters have a sensitive layer sandwiched between to heavy absorber layers
  - Absorber layers useful to create showers of secondary particles
- Useful for hadrons because a homogeneous detector would be too large
  - Worse resolution, though

---

iCSC 2014 24-25 February 2014, CERN
Jet clustering

Because we are measuring decay products we must find a way to cluster them together to accurately represent the original particle

- A few theoretical considerations
- **Infrared Safe**
  - Should not be sensitive to soft radiation
- **Collinear Safe**
  - Should not be sensitive to collinear radiation

Jet clustering

Because we are measuring decay products we must find a way to cluster them together to accurately represent the original particle

- A few theoretical considerations
- **Infrared Safe**
  - Should not be sensitive to soft radiation
- **Collinear Safe**
  - Should not be sensitive to collinear radiation

---

**Cone Algorithms**

- "Seed" defines approximate jet direction
- All energy deposits within a given radius are put into the jet
- The centroid is determined summing all particles within the cone
- The centroid becomes the new seed
  - Iterated until stable

---

**Mid-point cone algorithm**

- Search for missing jets using the midpoint of all jet pairs as a seed
- If there is a stable cone consider the energy deposits shared between the two jets ($E_{\text{Shared}}$)
- Take $f = E_{\text{Shared}}/E_{\text{Jet}_2}$
  - If $f > 50\%$ merge the jets; else split the jets

---

**K_T algorithm**

- Begin with a list of hits and calorimeter towers
- Calculate
  - For each precluster $i$: $d_i = \frac{p_{T,i}}{p_{T,j}}$
  - For each pair $(i,j)$: $d_{ij} = \min(p_{T,i}, p_{T,j}) \frac{\Delta y^2 + \Delta \phi^2}{D^2}$
- Find the minimum, $d_{\min}$, of all $d_i$ and $d_{ij}$
- If $d_{\min}$ is a $d_{ij}$, remove preclusters $i$ and $j$ from the list and replace with a new merged precluster
- If $d_{\min}$ is a $d_i$, precluster $i$ is not "mergeable" and can be added to the list of jets
- Repeat until list is exhausted
Some different Examples

Combining Measurements
- It can be advantageous to use different parts of the detectors for different measurements for the constituents of the jets.
- For example, we can sometime replace a calorimeter measurement with a tracker measurement associated to it.
  - For low momenta, the tracker measurements are more precise.

Particle ID - RICH
- Cherenkov radiation is emitted when a particle passes through a medium and is initially going faster than the speed of light in that medium.
- A ring of light is emitted that is proportional to the momentum.
- By choosing the correct media, we can use this as a for of particle identification:
  - Light in $C_1$ and $C_2$ = Pion
  - Light in just $C_1$ = Kaon
  - No light = proton
Multivariate Techniques

- Finally we can combine all these measurements into very powerful multivariate analysis (MVA) techniques
  - These can give a measure of how likely a jet is to be a b-jet
- One technique is a decision tree that makes a series of cuts on different input variables
  - Then reclassified by the Gini index $p(1-p)$

Decision trees - weighting

- The tree is trained against a known truth (from MC)
- Misclassified events are given a larger weight then retrained

Sample Result

- In the end we get a better discriminator than any simple cut

Inputs

- Tracks (ingredients)
- Leptons (electrons, muons)
- Impact parameters
- Secondary vertices (intermediate objects)
- Simple secondary vertex
- Soft muon
- Soft electron
Output

- Using the inputs from multiple parts of the detector, we can make a better judgement on if the jet we are measuring came from a b-quark or another source.

Conclusions

- Jet algorithms are designed to make up for the inefficiencies created by finite resolution of our calorimeters.
- Aspects from many different portions of the detector can be combined using statistical tools such as decision trees to determine how likely it is that a particular jet came from a b-quark.
- These algorithms are not as computing intensive and well suited for personal computing.
- Data structures must be manipulated in order for these to be run on laptops.
LECTURE 7
Read-Out Electronics: where data come from

Tuesday 25 February

Description
This lecture will focus on electronics used in HEP: while the software tools available to program very performant firmware are more and more powerful, hardware limitations cannot be (easily) overcome. High speed digital signals, today required in HEP setups, need to be handled as analog signals: signal propagation, interaction between signals and electromagnetic interferences must be carefully considered. Moreover, the power consuming of ASICs and FPGAs strongly depends of the operations required and may be a limitation of the board.

Audience
This lecture targets everyone interested in how the signals are transported from a detector to the data storage, which could be the hardware limitation of the available electronics and which characteristics must be taken into account in the choice of a good electronics for the experiment. After this lecture the attendees are expected to have acquired a good understanding of the hardware limitation of electronics and the characteristics needed to preserve the integrity of the signal to be stored for the analysis of an experiment.

Pre-requisite
Though all lectures are in principle independent, there might be some benefit for listeners to have attended the first lecture of the series.
Read-Out Electronics: where data come from

Lecture 2

Francesco Messi
Rheinische Friedrich-Wilhelms-Universität Bonn – DE

Outlook

- Prologue
- The magic key: Electronics
- The FPGA (Field Programmable Gate Array)
- An example: features extraction from a Sampling-ADC
- From the chip to the board
- High Speed Digital Signals

Prologue...

Modern experimental set-ups
- Modern experimental set-ups are made of a large number of different detectors, each looking at a particular aspect
- Modern detectors are producing analog electrical signals
- Detectors are “talking” through the electronics

Read-Out Electronics: where data come from

Lecture 2
Read-Out Electronics: where data come from (lecture 2)

Prologue…

Electronics as a “transmission line” from the detector to the data storage

- **We want:**
  - to receive the correct information → reliability
  - to receive the full information transmitted → efficiency
  - to understand the message → no distortion

- **Read-Out Electronics:**
  - From a little, fast analog signal to a bit-stream of data

---

From the analog to the digital world

- Digital signal → two states: present or absent, “1” or “0”

<table>
<thead>
<tr>
<th>Family</th>
<th>One (V)</th>
<th>Zero (V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTL</td>
<td>2 to 5</td>
<td>0 to 0.8</td>
</tr>
<tr>
<td>NIM</td>
<td>3 to 12</td>
<td>-2 to 1.5</td>
</tr>
<tr>
<td>ECL</td>
<td>-0.81 to -1.13</td>
<td>-1.48 to -1.95</td>
</tr>
<tr>
<td>LVDS</td>
<td>0.92 to 1.12</td>
<td>1.27 to 2.4</td>
</tr>
</tbody>
</table>

...and many more...

---

Prologue…

Binary logic: functions

- **AND**
- **OR**
- **XOR**
- **NOT**
- **NAND**
- **NOR**
- **NXOR**

---

Prologue…

Binary logic: FlipFlop

<table>
<thead>
<tr>
<th>CK</th>
<th>S</th>
<th>R</th>
<th>Qn</th>
<th>Qn+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
The magic key: electronics

what do we gain by the usage of electronics and what do we have to worry about?

1) It is much faster than human in taking decisions
   - e.g. evolution of photo cameras
     - automatic setting of focus, timing, opening...
   - Fast response
     - yesterday
     - today
   - Fast set-up
     - Much easier to duplicate data
     - Storage/management of lots of data more simple...

What do we gain from the usage of electronics?

2) Allows to trigger
   - Choice of record when a specific scenario happens
   - Why? We can not record all the information
     - Need to choose which one is interesting...
   - E.g.: bubble chamber experiments: cannot trigger =
     - One picture each spill
     - millions of pictures of the chamber
     - maybe few hundreds with an event...

Think about the LHC experiments:
   - even with selecting the event to be stored (= triggering), the data-stream to the storage system is of the order 100 Mb/s
   - Different from monitoring the temperature in a room:
     - not sampling at a fixed frequency,
     - but analyzing the event in real time and taking a decision...
The magic key: electronics

Triggering…

- E.g. a movie and a picture:
  - The video camera is recording everything:
    - you will not lose events, but...  
    - big amount of data to be stored
    - once you look out for the event, you need to watch all the movie
  - The photo camera is registering only scenes that are “triggered” by the operator:
    - less data to be stored
    - much simpler to look at them, but...
    - you lose all what happened in between one trigger and another

⇒ Be careful: acquired raw data are a selected subsection of what the experimental set-up could detect!!!
Signal processing: the FPGA

Some numbers…

<table>
<thead>
<tr>
<th></th>
<th>Spartan-6</th>
<th>Virtex-7</th>
<th>Virtex UltraScale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic Cells</td>
<td>147,443</td>
<td>1,954,560</td>
<td>4,407,480</td>
</tr>
<tr>
<td>BlockRAM</td>
<td>4.8Mb</td>
<td>68Mb</td>
<td>115Mb</td>
</tr>
<tr>
<td>DSP Slices</td>
<td>180</td>
<td>3,600</td>
<td>2,880</td>
</tr>
<tr>
<td>DSP Performance</td>
<td>140 GMACs</td>
<td>5,335 GMACs</td>
<td>4,268 GMACs</td>
</tr>
<tr>
<td>(symmetric FIR)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transceiver Count</td>
<td>8</td>
<td>96</td>
<td>104</td>
</tr>
<tr>
<td>Transceiver Speed</td>
<td>3.2 Gb/s</td>
<td>28.05 Gb/s</td>
<td>32.75 Gb/s</td>
</tr>
<tr>
<td>Total Transceiver</td>
<td>50 Gb/s</td>
<td>2,784 Gb/s</td>
<td>5,101 Gb/s</td>
</tr>
<tr>
<td>Bandwidth (full duplex)</td>
<td>800</td>
<td>1,866</td>
<td>2,400</td>
</tr>
<tr>
<td>Memory Interface (DDR3)</td>
<td>576</td>
<td>1,200</td>
<td>1,456</td>
</tr>
<tr>
<td>I/O Pins</td>
<td>1.2V - 3.3V</td>
<td>1.2V - 3.3V</td>
<td>1.0 – 3.3V</td>
</tr>
<tr>
<td>I/O Voltage</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


Why should an FPGA be used?

- Programmable = re-usage of the same hardware for multiple tasks
- Parallelism = in principle any n-to-one logic
  - Massive performances on same algorithms
- Exact determination of the execution order of the different tasks
  - Hardware chain of binary functions

An example…

Each detector used in experimental set-ups consists of numerous channels, detecting signals with rate of the order of MHz or even GHz: a large amount of data are produced. To select which data are interesting to be recorded, it is important to process the data as soon as possible near the detector…

An example

Feature extraction from a Sampling-ADC

- Baseline
- Slope
- Start time
- Charge integral
- Etc..

- Needs to be done as fast as possible
- With a fixed delay time
- In parallel for many channels
An example
Feature extraction from a Sampling-ADC

- Less cabling
- Less electronics modules
- Trigger capability
- But...
  - Sampling of the analog signal

From the chip to the board

Each electronic chip is generally mounted on boards, to provide the necessary power and buffering of the signals; even if the performance of the chip itself (FPGA or ASIC) is very high, one of the other components can limit the characteristic of the board...
From the chip to the board

- FPGA: very powerful
  …ok but what about the electronics around it???
  - Not possible to connect the signals from the detector directly to the FPGA!
    - Your transmission line could “speak a different language”
    - Power and filtering are needed
    - Space constrains
    - BUFFERING:
      - never trust the transmission lines: unknown signals can arrive and we don’t want to damage our FPGA chips!!

From the chip to the board

- Not always the most performing board is the correct choice:
  - more performance = more power
  - more power = more cooling
  - more cooling = more material budget in the detector…
  - Is it fine with “my physics”???
  - It is important to look at the full system, more than at the single module…

Signal processing

- FPGAs used for fast, parallel, on-line operations
- Not possible to connect analog signals directly to the FPGA!
- Bandwidth (in/out-put buffers, etc…)

System optimization more than component optimization

- Logistic constrains (space, cooling, power, etc…)
High Speed digital signals need to be handled as analogue ones…

- **A digital signal is in binary logic**
  - I have to take care only of two states: “1” and “0”
  - It is less affected by noise than an analog signal
  - Than it is much simpler to handle…
  - → **true in principle, but…**

---

Summary and conclusions

**Lecture 1**

- To investigate the matter, we build detectors
- Modern detectors are providing analog electrical signals
- Electronics is needed to “read the information” from the detectors
- The electronics chain has to:
  - Preprocess the signals w/o distortion
  - Be reliable
  - Be efficient
  - Be fast
  - Be calibrated to the experiment!!!

---

Read-Out Electronics: where data come from

**lecture 2 summary and conclusions**

- Modern electronics is very powerful
  - High bandwidth
  - High programmability
  - But…
- One element is enough to decrease the performance of the full chain
  - → System optimization more than component optimization
- High speed digital signals need to be handled as analog ones
LECTURE 8
Introduction to machine learning and data mining

Tuesday 25 February

Description
In this lecture, the basis of machine learning and data mining will be explained. Then, the most typical problems where machine learning is applied will be presented: classification, clustering, regression and anomaly detection. For those problems, some techniques that can be applied will be presented and briefly explained: decision trees, support machine vectors, k-NN, k-means and neural networks.

Audience
After this lecture, the attendees are expected to understand the differences between machine learning and data mining, comprehend the general concepts of machine learning as well as some of the typical problems and its solution approaches.

Pre-requisite
This lecture can be followed by anyone having experience with algorithms and computer science.

Juan Lopez Gonzalez
CERN
1. Introduction

1.1. Some definitions
1.2. Machine learning vs Data mining
1.3. Examples
1.4. Essence of machine learning
1.5. A learning puzzle
1.1 Some definitions

- **To learn**
  - To use a set of observations to uncover an underlying process

- **To memorize**
  - To commit to memory
  - *It doesn’t mean to understand*

1.2 Machine learning vs Data mining

- **Machine learning** (Arthur Samuel)
  - Study, design and development of algorithms that give computers capability to learn without being explicitly programmed.

- **Data mining**
  - Extract knowledge or unknown patterns from data.

1.3 Examples

- **Credit approval**
  - Gender, age, salary, years in job, current debt…

- **Spam filtering**
  - Subject, From…

- **Topic spotting**
  - Categorize articles

- **Weather prediction**
  - Wind, humidity, temperature…

1.4 Essence of machine learning

- A pattern exists
- We cannot pin it down mathematically
- We have data on it
1.5 A learning puzzle

2. Definition

2.1. Components
2.2. Generalization and representation
2.3. Types of learning

2.1 Components

- **Input** (customer application)
- **Output** (approve/reject credit)
- **Ideal function** ($f: X \mapsto Y$)
  - Data: $(a_1, b_1, \ldots, n_1), (a_2, b_2, \ldots, n_2), \ldots, (a_N, b_N, \ldots, n_N)$ (historical records)
  - Result: $(y_1), (y_2), \ldots, (y_N)$ (loan paid or not paid)
- **Hypothesis** ($g: X \mapsto Y$)
2.2 Generalization and representation

- **Generalization**
  - The algorithm has to build a general model
- **Objective**
  - Generalize from experience
  - Ability to perform accurately for unseen examples
- **Representation**
  - Results depend on input
  - Input depends on representation
    - Pre-processing?

2.3 Types of learning

- **Supervised**
  - Input and output
- **Unsupervised**
  - Only input
- **Reinforcement**
  - Input, output and grade of output

3. Problems

- 3.1. Regression
- 3.2. Classification
- 3.3. Clustering
- 3.4. Association rules

3.1 Regression

- Statistical process for estimating the relationships among variables
  - Could be used for prediction
3.2 Classification

- Identify to which of a set of categories a new observation belongs
  - Supervised learning

3.3 Clustering

- Grouping a set of objects in such a way that objects in the same group are more similar
  - Unsupervised learning

3.4 Association rule

- Discovering relations between variables in large databases
  - Based on ‘strong rules’
  - If order matters -> Sequential pattern mining

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

4. Techniques

4.1 Decision trees
4.2 SVM
4.3 Monte Carlo
4.4 K-NN
4.5 ANN
4.1 Decision trees

- Uses tree-like graph of decisions and possible consequences
  - Internal node: attribute
  - Leaf: result

- Results human readable
- Easily combined with other techniques
  - Possible scenarios can be added
- Expensive

4.2 Support Vector Machine (SVM)

- Separates the graphical representation of the input points
  - Constructs a hyperplane which can be used for classification
    - Input space transformation helps
  - Non-human readable results

4.3 Monte Carlo

- Obtain the distribution of an unknown probabilistic entity
  - Random sampling to obtain numerical results

- Applications
  - Physics
  - Microelectronics
  - Geostatistics
  - Computational biology
  - Computer graphics
  - Games
  - …
4.4 K-Nearest neighbors (K-NN)

- Classifies by getting the class of the K closest training examples in the feature space

K=1  
K=5

4.4 K-Nearest neighbors (K-NN)

- Easy to implement
- naive version
- High dimensional data needs dimension reduction
- Large datasets make it computational expensive
- Many k-NN algorithms try to reduce the number of distance evaluations performed

4.5 Artificial neural networks (ANN)

- Systems of interconnected neurons that compute from inputs

Human | Artificial
---|---
Neuron | Processing element
Dendrites | Combining function
Cell body | Transfer function
Axons | Element output
Synapses | Weights

Human neuron | Artificial neuron
4.5 Artificial neural networks (ANN)

Example:

- **Perceptron**
  - Single-layer artificial network with one neuron
  - Calculates the linear combination of its inputs and passes it through a threshold activation function
  
  \[
  y = \sigma \left( \sum_{i=1}^{2} w_i x_i + \theta \right)
  \]
  
  \[\sigma(x) = \begin{cases} 
  1 & \text{if } x > 0 \\
  -1 & \text{otherwise}.
  \end{cases}\]

  Equivalent to a linear discriminant

- **Learning**
  - Learn the weights (and threshold)
  - Samples are presented
    - If output is incorrect adjust the weights and threshold towards desired output
    - If the output is correct, do nothing
Q & A
LECTURE 9
Self organizing maps
A visualization technique with data dimension reduction

Tuesday 25 February

<table>
<thead>
<tr>
<th>16:00-17:00</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In this lecture, the general concepts of self-organizing maps and its properties will be explained. Starting from the classic neural network approach, a MLP (introduced in the previous lecture), the concept of SOM will be explained. Its structure, the learning process and the later classification of the inputs for not seen cases. The main features of the maps: dimensional reduction and the conservation of the topological properties of the inputs, will be highlighted. Also, a small example will be shown where the attendants will see an actual map arranging itself and the resultant order will be interpreted. Finally, some other SOM based models will be shown to point out different architectures and possibilities.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Audience</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>After this lecture, the attendees are expected to understand how basic self-organizing maps are built, to understand and interpret the properties of the resultant maps. Also, they will learn its advantages and disadvantages. Finally, they will see more complex SOM structures that could help them creating more specific problem-oriented models.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Pre-requisite</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This lecture can be followed by anyone having a basic knowledge of Artificial Neural Networks and machine learning.</td>
</tr>
</tbody>
</table>

Juan Lopez Gonzalez
CERN
Self organizing maps
A visualization technique with data dimension reduction

Juan López González
University of Oviedo

Inverted CERN School of Computing, 24-25 February 2014

LECTURE 1

Self organizing maps.
A visualization technique with data dimension reduction.

1. Artificial neural networks (ANN)

Lecture 2

• Lecture 1
  • Machine learning
    • Introduction
    • Definition
    • Problems
    • Techniques

• Lecture 2
  • ANN introduction
  • SOM
  • Simulation
  • SOM based models
1.1. Introduction

1.2. Types

1.2.1. Feedforward NN
1.2.2. Recurrent NN
1.2.3. Self organizing NN
1.2.4. Others

1.2.1. Feedforward NN

- Single layer feedforward
1.2.1. Feedforward NN

- Multi-layer feedforward
  - Supervised learning
  - Backpropagation learning algorithm

1.2.2. Recurrent neural networks

- Elman networks
  - ‘Context units’
  - Maintain state

- Hopfield network
  - Symmetric connections
  - Associative memory
1.2.2. Recurrent neural networks

- Modular neural networks
  - The human brain is not a massive network but a collection of small networks

1.2.3. Self-organizing networks

- Self-organizing networks
  - A set of neurons learn to map points in an input space to coordinates in an output space

1.2.4. Others

- Holographic associative memory
- Instantaneously trained networks
- Learning vector quantization
- Neuro-fuzzy networks
- ...

2. Self-organizing maps

- 2.1. Motivation
- 2.2. Goal
- 2.3. Main properties
- 2.4. Elements
- 2.5. Algorithm
2.1. Motivation

- **Topographic maps**
  - Different sensory inputs (motor, visual, auditory...) are mapped in areas of the cerebral cortex in an orderly fashion.

2.2. Goal

- Transform incoming signal of arbitrary dimension into a 1-2-3 dimensional discrete map in a topologically ordered fashion.

2.3. Main properties

- Transform continuos input space to discrete output space
  - **Dimension reduction**
    - winner-takes-all neuron
  - Ordered feature map

  *Input with similar characteristics produce similar output*
### 2.3.1 Dimension reduction

- **Curse of dimensionality** (Richard E. Bellman)
  - The amount of data needed grows exponentially with the dimensionality

- **Types**
  - **Feature extraction**
    - Reduce input data (features vector)
  - **Feature selection**
    - Select subset (remove redundant and irrelevant data)

### 2.4. Elements

- **...of machine learning**
  - A pattern exists
  - We don’t know how to solve it mathematically
  - A lot of data
    - \((a_1, b_1, ..., n_1)\), \((a_2, b_2, ..., n_2)\) ... \((a_N, b_N, ..., n_N)\)

- **Lattice of neurons**
  - Size?
  - Weights

- **Learning rate**
- **Neighborhood function**

- **Learning rate function**
- **Neighbourhood function or Mexican Hat function**
2.5. Algorithm

- Initialization
  - Input data preprocessing
    - Normalizing
    - Discrete-continuous variables?
  - Weight initialization
    - Random weights

2.5. Algorithm (2)

- Sampling
  - Take sample from input space
- Matching
  - Find BMU: i.e. min of $\sum_{i=1}^{D}(x_i - w_{ji})^2$
- Update weights
  - i.e. $\Delta w_{ji} = \eta(t) T_{i,j}(t)(x_i - w_{ji})$

3. Practical exercise

4. Map examples

  4.1. Digit recognition
  4.2. Finish phonetics
  4.3. Semantic map of word context
4.1. Digit recognition

4.2. Finnish phonetics

4.3. Semantic map of word context

5. Other SOM based models

5.1. TASOM
5.2. GSOM
5.3. MuSOM
5.1. TASOM

- Time adaptative self-organizing maps
- Deals with non-stationary input distributions
- Adaptative learning rates: $n(w,x)$
- Adaptative neighborhood rates: $T(w,x)$

5.2. GSOM

- Growing self-organizing maps
- Deals with identifying sizes for SOMs
  - Spread factor
  - New nodes in boundaries
- Good when unknown clusters

5.3. MuSOM

- Multimodal SOM
- High level classification from sensory integration

Q & A