



**ALICE**



ALFA - a common concurrency  
framework for ALICE and FAIR  
experiments

Mohammad Al-Turany  
GSI-IT/CERN-PH

# ALFA: New ALICE-FAIR framework

- Motivation for ALFA
- ALFA and FairRoot
- Transport layer
- Serializing message data
- Deployment system
- Examples
- Summary

# How it comes?

- FairRoot (2013)
  - Concurrency is an issue
  - Online and offline can be joined
- ALICE O<sup>2</sup> Project started (2013)
  - DAQ, Online and Offline : Together on one Framework

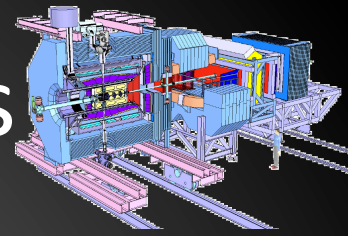


ALICE

# Similar online system requirements by ALICE and FAIR experiments

# Online System Requirements

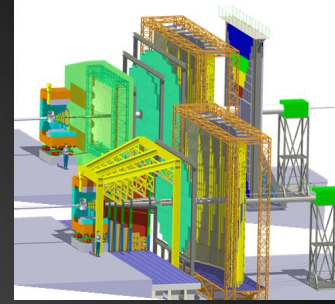
## PANDA



- Interaction rates of 20 MHz (50 MHz peak)
- Event size of 4-10 kB
- data rates after front end preprocessing:  
80GB/s - 300 GB/s
- No hardware trigger:
  - Autonomous recording frontends
  - Event selection in compute nodes

# Online System Requirements

## CBM



- At **10 MHz**, online data reduction by **1000** is mandatory (**1 TB/s**)
- Trigger signatures are complex (open charm) and require partial event reconstruction
- No a-priori association of signals to physical events!
- Need extremely fast reconstruction algorithms!

# Online System Requirements

## ALICE Run3:



- Sample full **50kHz Pb-Pb** interaction rate (current limit at  $\sim 500\text{Hz}$ , factor 100 increase)
- $\Rightarrow$   **$\sim 1.1$  TByte/s** detector readout but the ALICE data processing power capacity will not allow more than a Storage bandwidth **of  $\sim 20$  GByte/s**



# Strategy

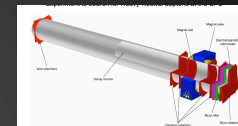
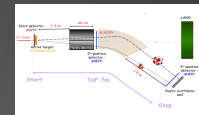
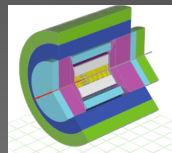
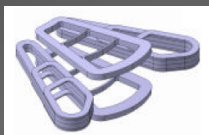
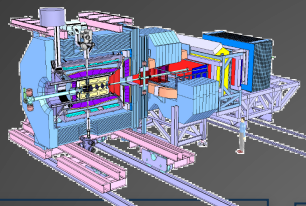


- Massive data volume reduction
  - Data reduction by (partial) online reconstruction and compression
- Much tighter coupling between online and offline reconstruction software





# FairRoot



Start testing the VMC concept for CBM

Panda decided to join-> FairRoot: same Base package for different experiments

R3B joined

EIC (Electron Ion Collider BNL)  
EICRoot

SOFIA (Studies On Fission with Aladin)

SHIP - Search for Hidden Particles

2004

2006

2010

2011

2012

2013

2014

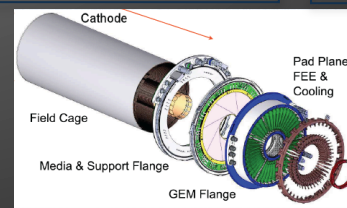
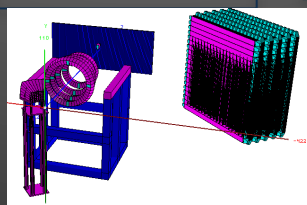
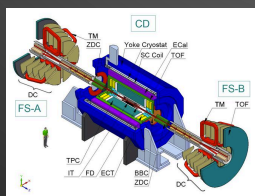
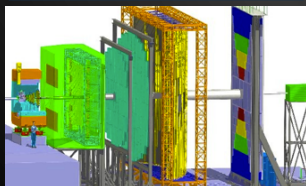
First Release of CbmRoot

MPD (NICA) start also using FairRoot

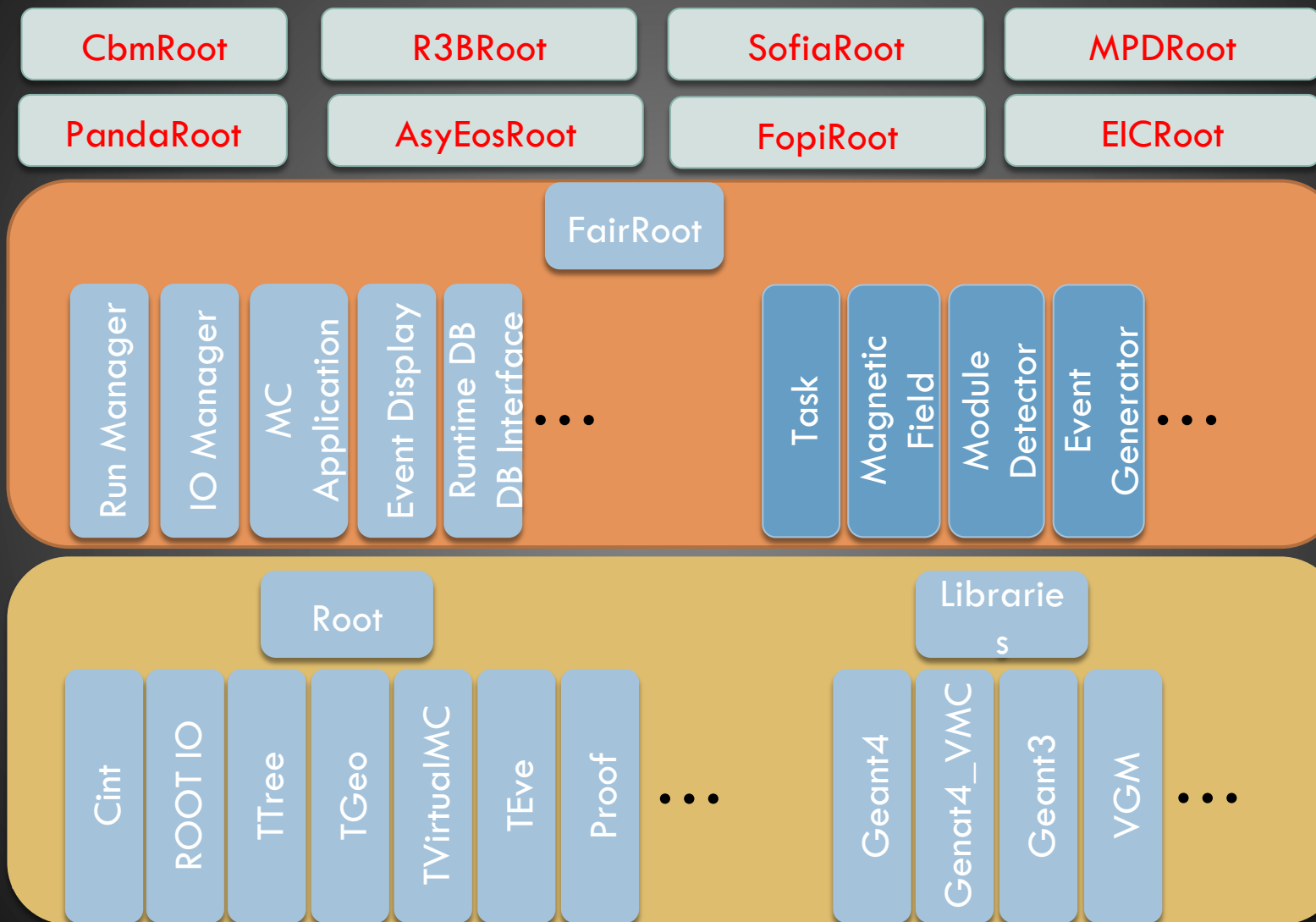
ASYEOS joined (ASYEOSRoot)

GEM-TPC separated from PANDA branch (FOPIRoot)

ENSAR-ROOT  
Collection of modules used by structural nuclear physics exp.



# Current status of FairRoot



# ALICE and FAIR: Why?

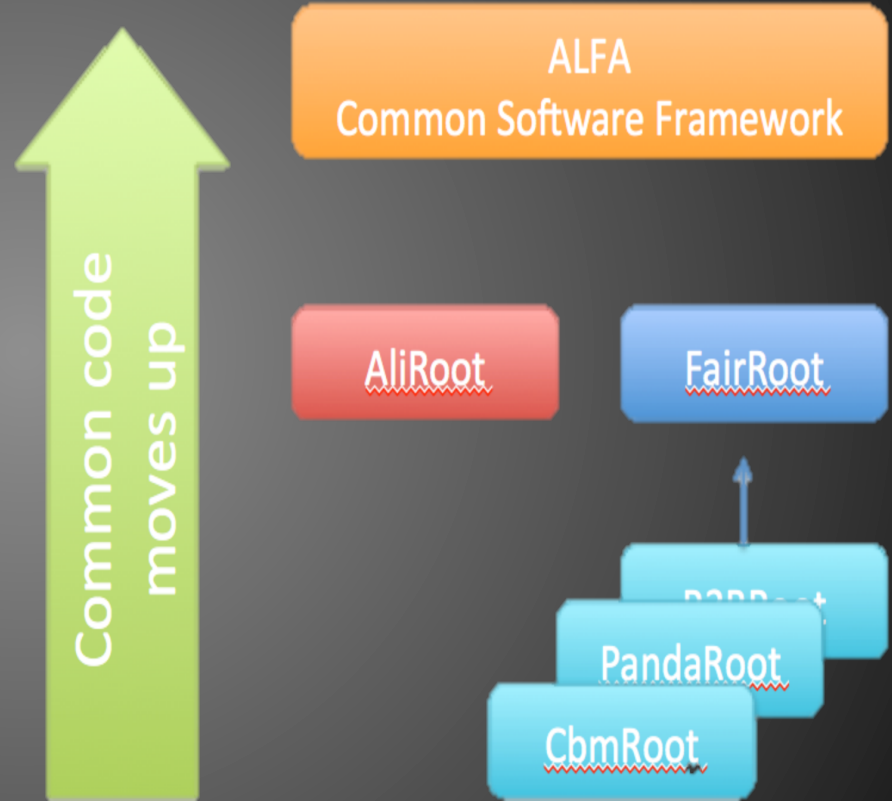
- A common framework will be beneficial for the FAIR experiments since it will be tested with real data and existing detectors before the start of the FAIR facility.
  - E.g.: Concepts for online calibrations and alignment can be tested in a real environment, similar to that of the planned FAIR experiments.
- ALICE will benefit from the work already performed by the FairRoot team concerning already implemented features (e.g. the continuous read-out, building and testing system, etc)

# Who is going to collaborate on ALFA

- ALICE DAQ
- ALICE HLT
- ALICE Offline
- FairRoot
- Fair and Non-Fair experiments are welcome to join

# ALFA

- Common layer for parallel processing.
- Common algorithms for data processing.
- Common treatment of conditions database.
- Common deployment and monitoring infrastructure.



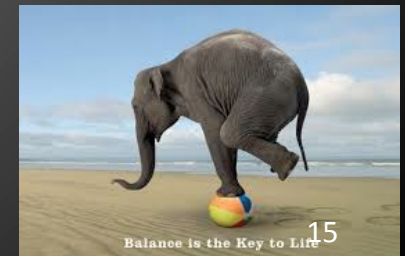
# ALFA



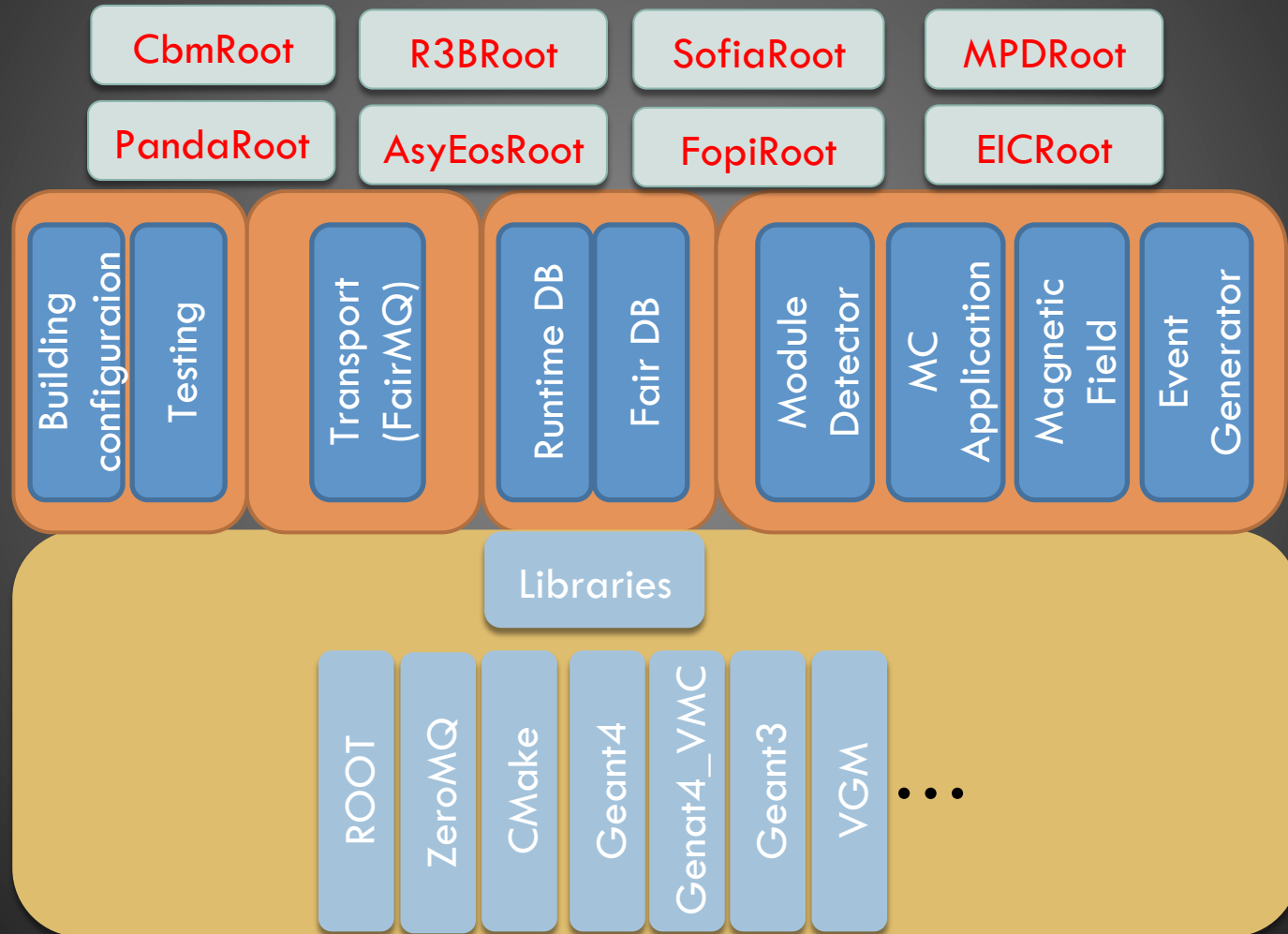
- Will rely on a data-flow based model (Message Queues).
- It will contain
  - Transport layer (based on: ZeroMQ, NanoMSG)
  - Configuration tools
  - Management and monitoring tools
- Provide unified access to configuration parameters and databases.
- It will include support for a heterogeneous and distributed computing system.
- Incorporate common data processing components

# Correct balance between reliability and performance

- Multi-process concept with message queues for data exchange
  - Each "Task" is a separate process, which can be also multithreaded, and the data exchange between the different tasks is done via messages.
  - Different topologies of tasks that can be adapted to the problem itself, and the hardware capabilities.



# FairRoot is undergoing a re-design process to make it more modular





# How is it with ALFA and FairRoot?

AliRoot6  
(O<sup>2</sup>)

CbmRoot

R3BRoot

SofiaRoot

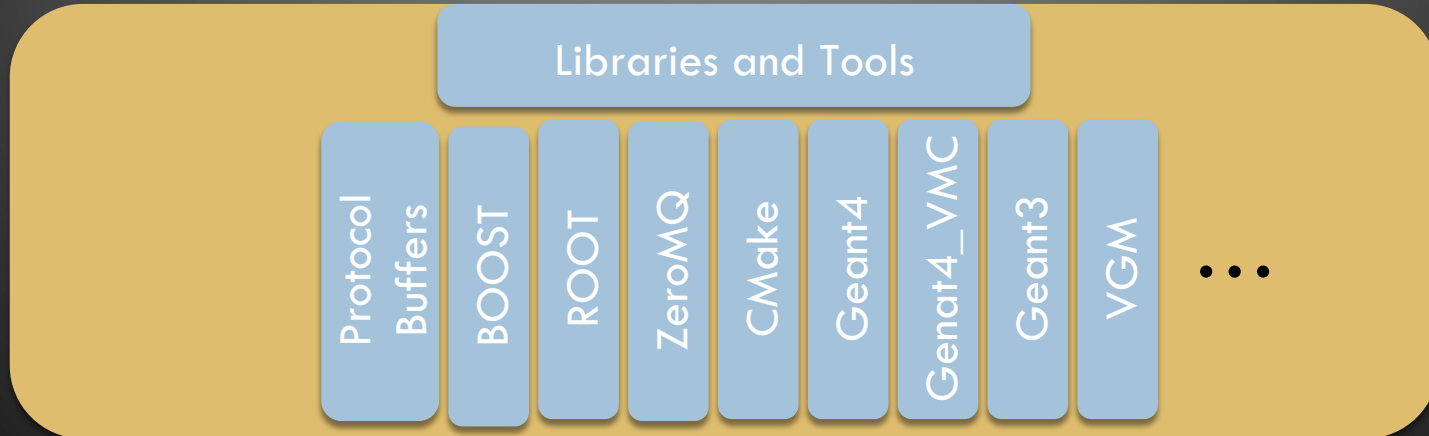
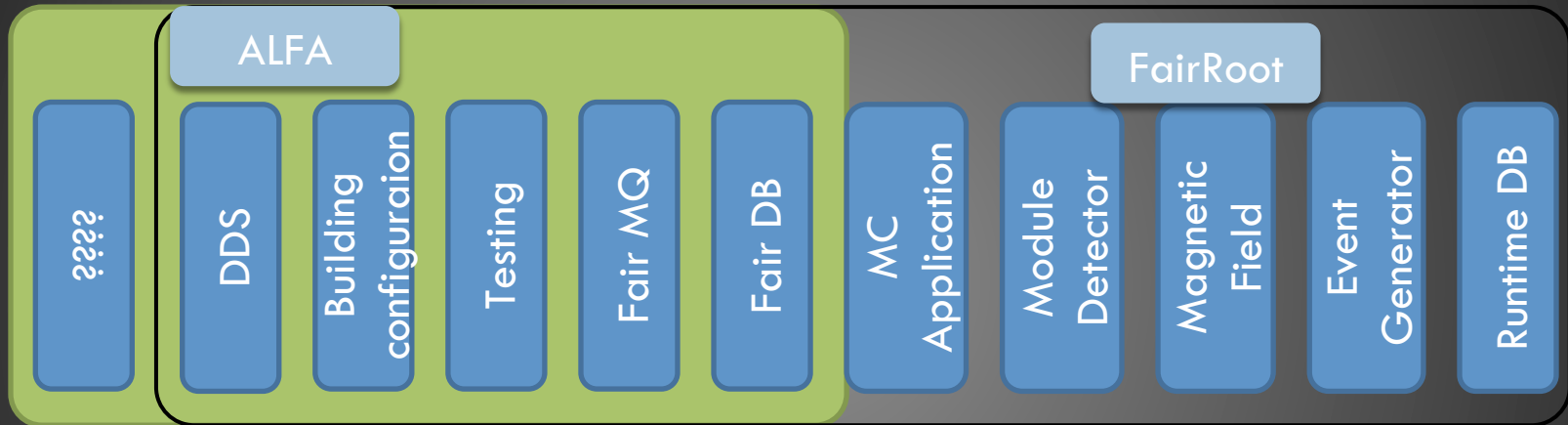
MPDRoot

PandaRoot

AsyEosRoot

FopiRoot

EICRoot



# ALFA will use ZMQ to connect different pieces together

- BSD sockets API
- Bindings for 30+ languages
- Lockless and Fast
- Automatic re-connection
- Multiplexed I/O

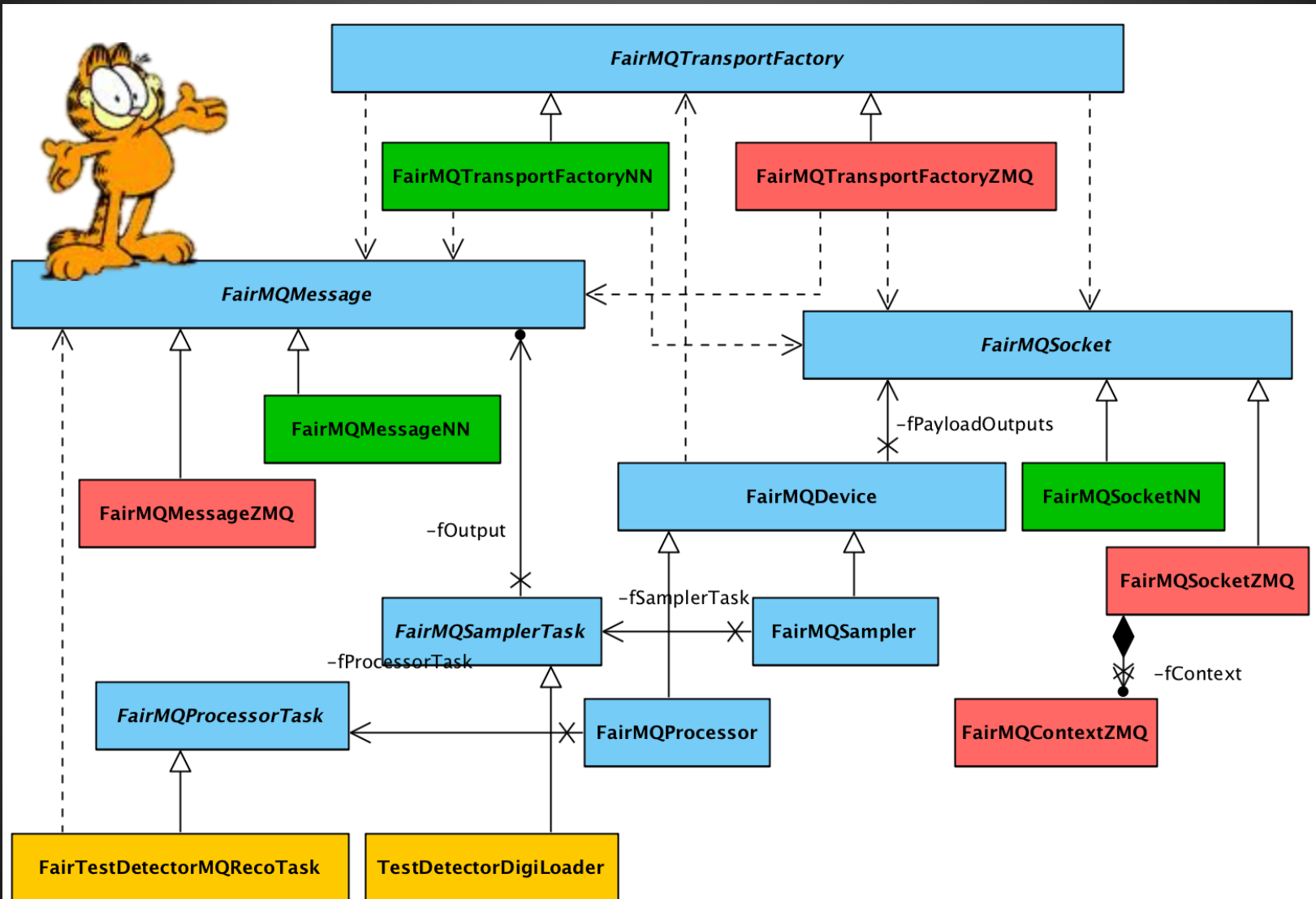


# BUT: nanomsg is under development by the original author of ZeroMQ

The logo for nanomsg, consisting of the word "nanomsg" in a bold, lowercase, sans-serif font. The text is white and is set against a solid black rectangular background.

- Pluggable Transports:
  - ZeroMQ has no formal API for adding new transports (Infiniband, WebSockets, etc). nanomsg defines such API, which simplifies implementation of new transports.
- Zero-Copy:
  - Better zero-copy support with RDMA and shared memory, which will improve transfer rates for larger data for inter-process communication.
- Simpler interface:
  - simplifies some zeromq concepts and API, for example, it no longer needs Context class.
- Numerous other improvements, described here:  
<http://nanomsg.org/documentation-zeromq.html>
- FairRoot is independent from the transport library
  - Modular/Pluggable/Switchable transport libraries.

# ALFA has an abstract transport layer



# How the different processes are going to communicate with each other?

- Different languages and hardware are possible how should the data exchange be done?
- Schema evolution?

# Protocol buffers

- Google Protocol Buffers support is now implemented
  - Example in Tutorial 3 in FairRoot.
- To use protobuf, run cmake as follows:
  - `cmake -DUSE_PROTOBUF=1`

# Boost serialization

- Code portability - depend only on ANSI C++ facilities.
- Code economy - exploit features of C++ such as RTTI, templates, and multiple inheritance, etc. where appropriate to make code shorter and simpler to use.
- Independent versioning for each class definition. That is, when a class definition changed, older files can still be imported to the new version of the class.
- Deep pointer save and restore. That is, save and restore of pointers saves and restores the data pointed to.
- ....

[http://www.boost.org/doc/libs/1\\_55\\_0/libs/serialization/doc/index.html](http://www.boost.org/doc/libs/1_55_0/libs/serialization/doc/index.html)

This is used already by the CBM Online group in Frankfurt and to we need it to exchange data with them!

# Protobuf, Boost or Manual serialization?

- Boost:
  - we are generic in the tasks but intrusive in the data classes (digi, hit, timestamp)
- Manual and Protobuf
  - we are generic in the class but intrusive in the tasks (need to fill/access payloads from class with set/get x, y, z etc ).

Manual method is still the fastest, protobuf is 20% slower and boost is 30% slower.

preliminary



# How to deploy ALFA on a laptop, few PCs or a cluster?

- We need to utilize any RMS (Resource Management system)
- We should also run with out RMS
- Support different topologies and process dependencies



# How to deploy ALFA on a laptop, few PCs or a cluster?

- We have to develop a dynamic deployment system (DDS):
  - An independent set of utilities and interfaces, which provide a dynamic distribution of different user processes by any given topology on any RMS.

# The Dynamic Deployment System (DDS) Should:

- Deploy task or set of tasks
- Use (utilize) any RMS (Slurm, Grid Engine, ... ),
- Secure execution of nodes (watchdog),
- Support different topologies and task dependencies
- Support a central log engine
- ....

See Talk by Anar Manafov on Alice Offline week (March 2014)  
<https://indico.cern.ch/event/305441/>

# Elements of the topology

## #### Task

- \* A task is a single entity of the system.
- \* A task can be an executable or a script.
- \* A task is defined by a user with a set of props and rules.
- \* Each task will have a dedicated DDS watchdog process.

## #### Collection

- \* A set of tasks that have to be executed on the same physical computing node.

## #### Group

- \* A container for tasks and collections.
- \* Only main group can contain other groups.
- \* Only group define multiplication factor for all its daughter elements.

MAIN

TASK\_1

TASK\_5

COLLECTION\_1

TASK\_1

TASK\_2

TASK\_3

TASK\_4

GROUP\_1  
N=10

TASK\_5

TASK\_6

COLLECTION\_2

TASK\_2

TASK\_7

# Current Status

- The Framework delivers some components which can be connected to each other in order to construct a processing pipeline(s).
- All components share a common base called Device
- Devices are grouped by three categories:
  - **Source:**
    - Data Readers (Simulated, raw)
  - **Message-based Processor:**
    - Sink, Splitter, Merger, Buffer, Proxy
  - **Content-based Processor:**
    - Processor



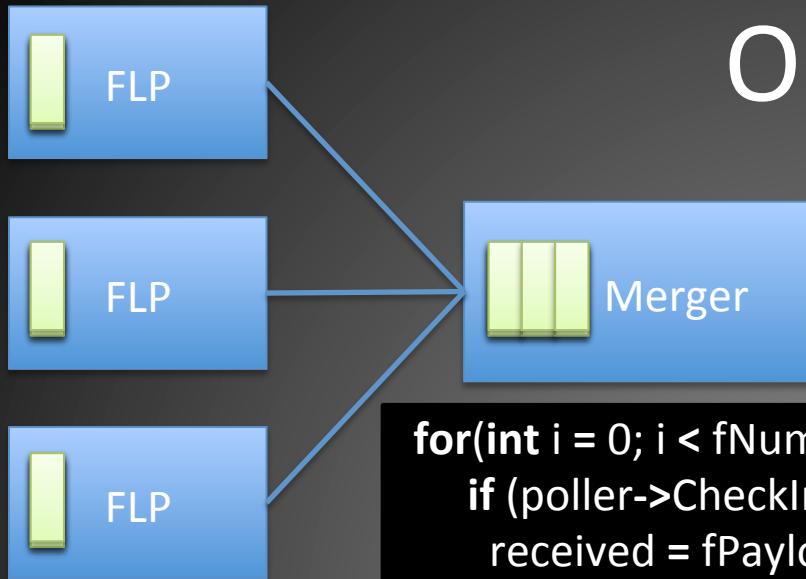
# Using multi-part messages

- It allows us to concatenate multiple messages into a single message without copying
- All parts of the message are treated as a **single atomic unit** of transfer, however the **boundaries** between message parts are **strictly preserved**
- A multi-part message is a message that has more than one frame. ZeroMQ sends this message as a single on-the-wire message and guarantees to **deliver all** the message parts (one or more), or **none** of them.





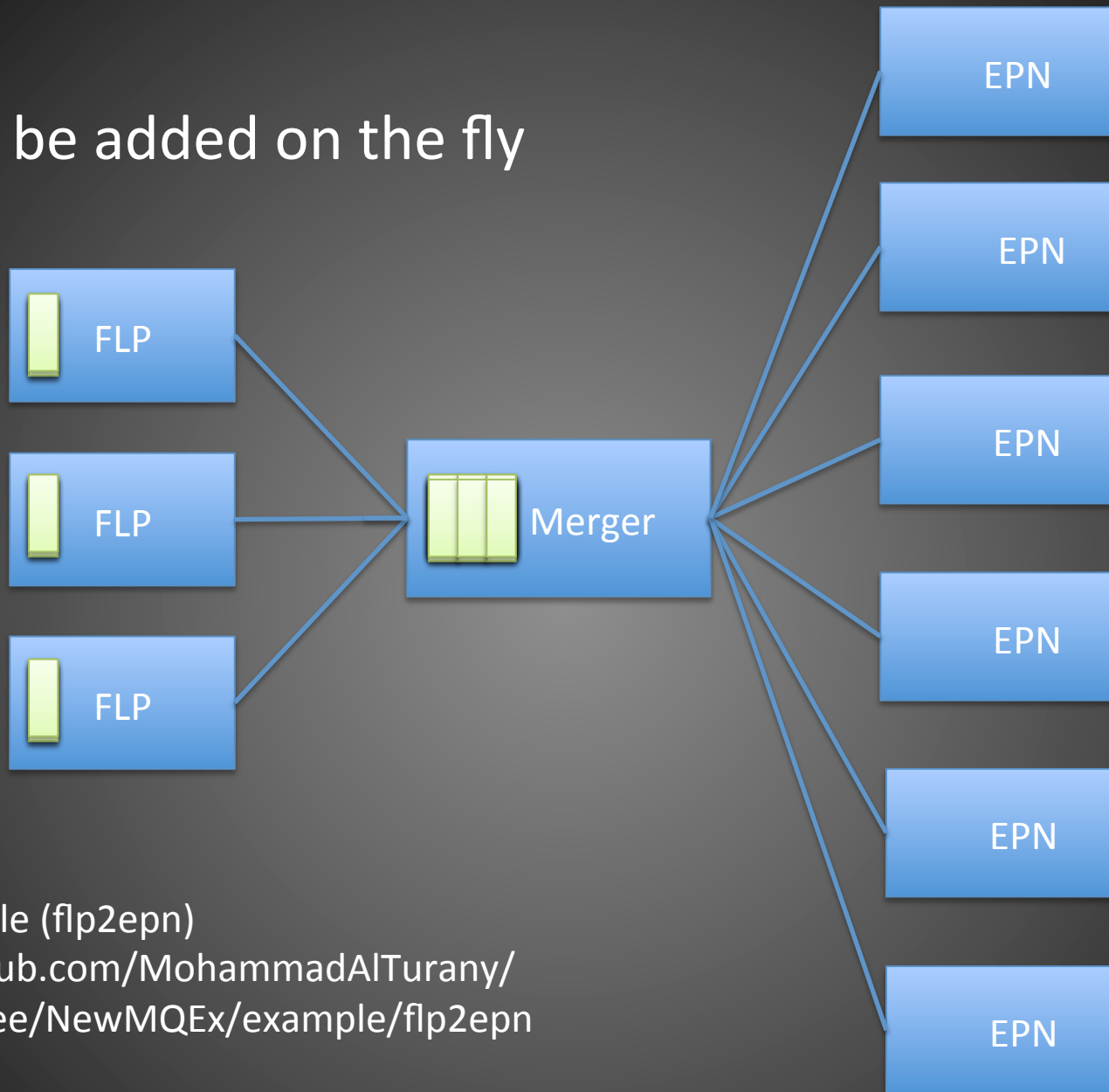
# O2Merger



```
for(int i = 0; i < fNumInputs; i++) {  
    if (poller->CheckInput(i)){  
        received = fPayloadInputs->at(i)->Receive(msg);  
    }  
    if (received) {  
        if(i<NoOfMsgParts){  
            fPayloadOutputs->at(0)->Send(msg, ZMQ_SNDMORE);  
        }else{  
            fPayloadOutputs->at(0)->Send(msg);  
        }  
        received = false;  
    }  
}
```

"Hold on! There are more data going to be added to this message!"

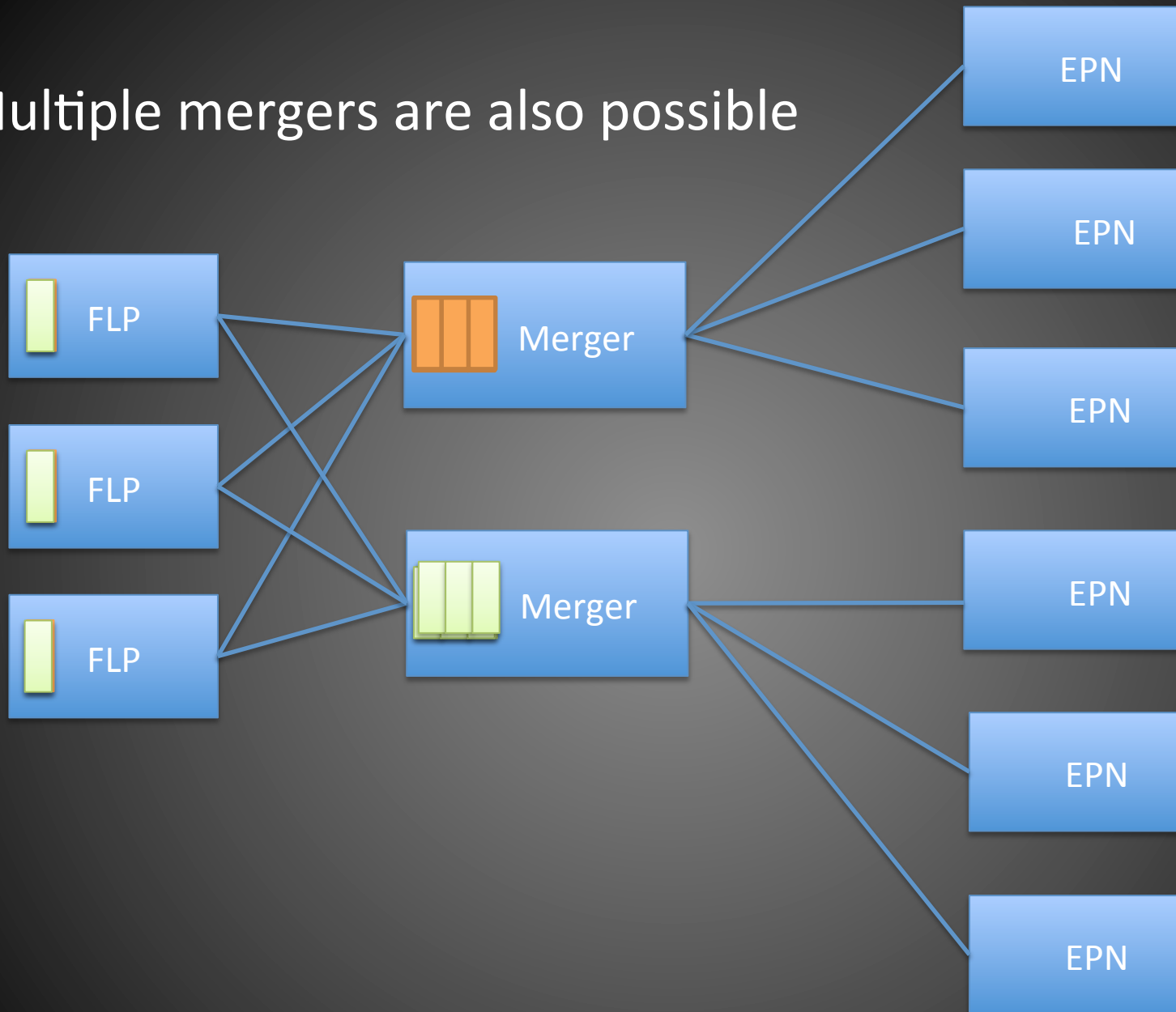
# EPN can be added on the fly



New example (flp2epn)

<https://github.com/MohammadALTurany/FairRoot/tree/NewMQEx/example/flp2epn>

# Multiple mergers are also possible



# Summary

- The ALFA project is starting
- The Communication layer is ready to use
- Different data serialization methods are available
- DDS and Topology parser are under development

# Backup and Discussion

# STORM is a very attractive option but no native support for C++ !

Storm was designed from the ground up to be usable with any programming language. At the core of Storm is a [Thrift definition](#) for defining and submitting topologies. Since Thrift can be used in any language, topologies can be defined and submitted from any language.

Similarly, spouts and bolts can be defined in any language. Non-JVM spouts and bolts communicate to Storm over a [JSON-based protocol](#) over stdin/stdout. Adapters that implement this protocol exist for [Ruby](#), [Python](#), [Javascript](#), [Perl](#), and [PHP](#).

*storm-starter* has an [example topology](#) that implements one of the bolts in Python.

<http://storm.incubator.apache.org/about/multi-language.html>

# The built-in core ØMQ patterns are:

- **Request-reply**, which connects a set of clients to a set of services. (remote procedure call and task distribution pattern)
- **Publish-subscribe**, which connects a set of publishers to a set of subscribers. (data distribution pattern)
- **Pipeline**, which connects nodes in a fan-out / fan-in pattern that can have multiple steps, and loops. (Parallel task distribution and collection pattern)
- **Exclusive pair**, which connect two sockets exclusively

