Recent FairRoot developments

Radoslaw Karabowicz for the FairRoot Group, GSI

PANDA L. Collaboration Meeting, 10.09.2014

FairSoft - externals

- moved to git (https://github.com/FairRootGroup/FairSoft) in mid 2014.
- latest release: jul14p3.
- SVN - closed, last release: dec13.
- PANDA officially uses apr13.
- new installation option for FairSoft (only CMake, boost, ROOT, ZeroMQ, gtest, gsl are build):

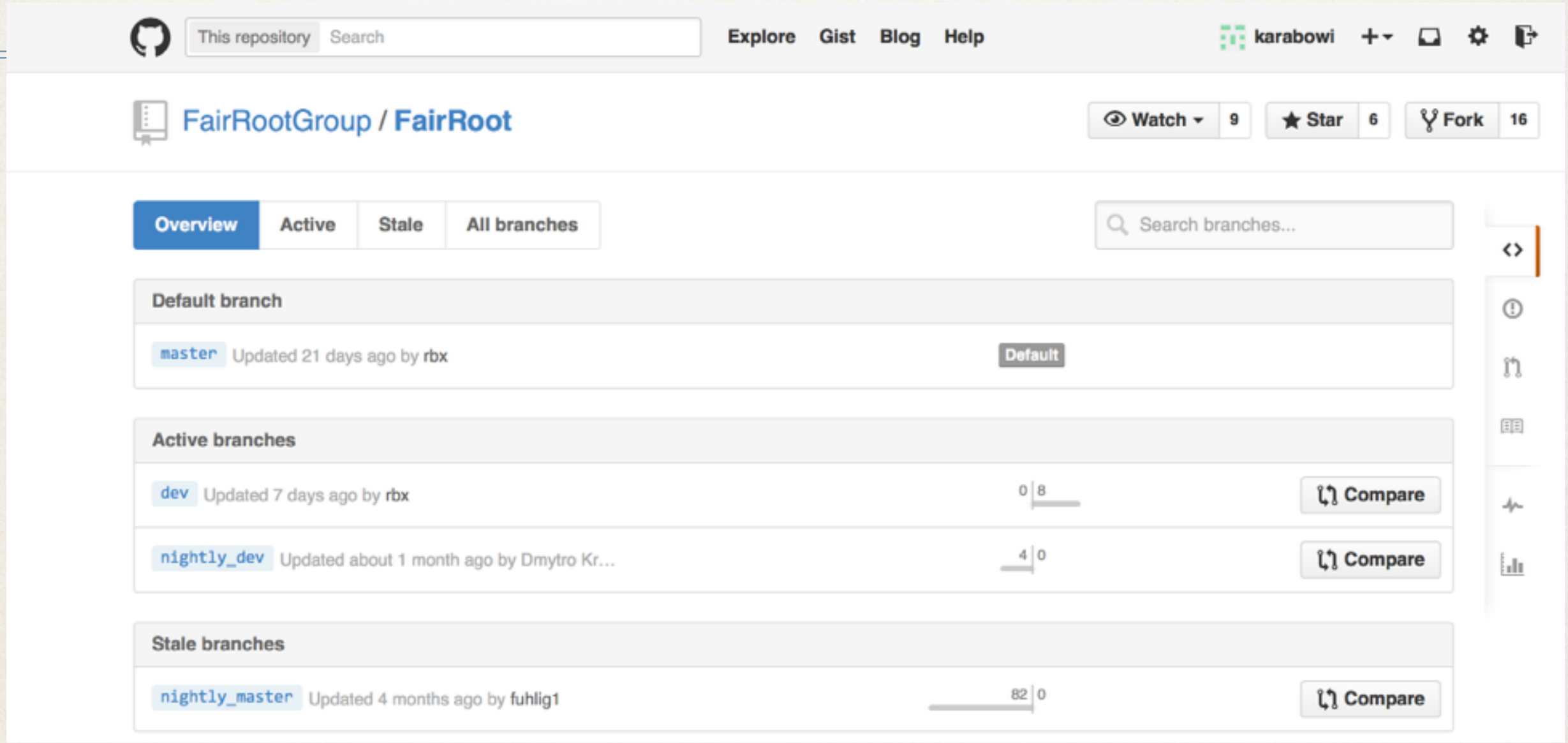
```
./configure mqonly
```

FairRoot

- moved to git (https://github.com/FairRootGroup/FairRoot) early in 2014.
- no more development in SVN (last commit 23489 in February 2014, last release: v-13.12).
- FairRoot is distributed under the terms of the GNU Lesser General Public License (LGPL) version 3.
- new installation option for FairRoot (exclude simulation and event generators):

```
cmake -DRECO_ONLY=1
```

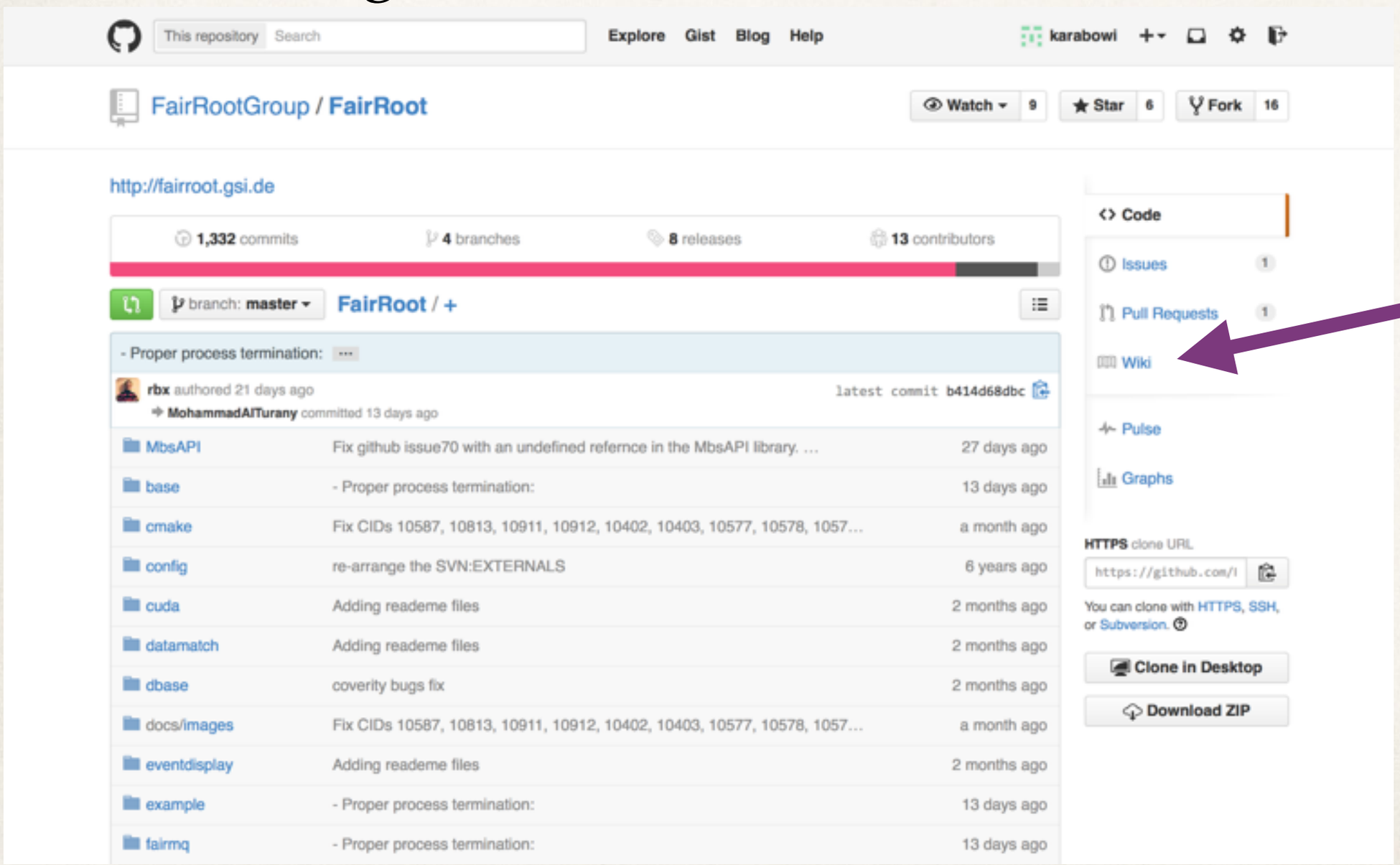
FairRoot branches



- many more in developers' repositories on github

documentation

- WIKI on github



documentation

- added README.md files in (almost) each FairRoot directory - it is automatically converted by github into easy-to-read explanatory pages:

README.md add the template usage description to the README 2 months ago

fairDoxy.conf add doxygen config 3 years ago

README.md

FairRoot

The framework

A simulation, reconstruction and analysis framework that is based on the ROOT system. The user can create simulated data and/or perform analysis with the same framework. Geant3 and Geant4 transport engines are supported, however the user code that creates simulated data do not depend on a particular monte carlo engine. The framework delivers base classes which enable the users to construct their detectors and /or analysis tasks in a simple way, it also delivers some general functionality like track visualization. Moreover an interface for reading magnetic field maps is also implemented.

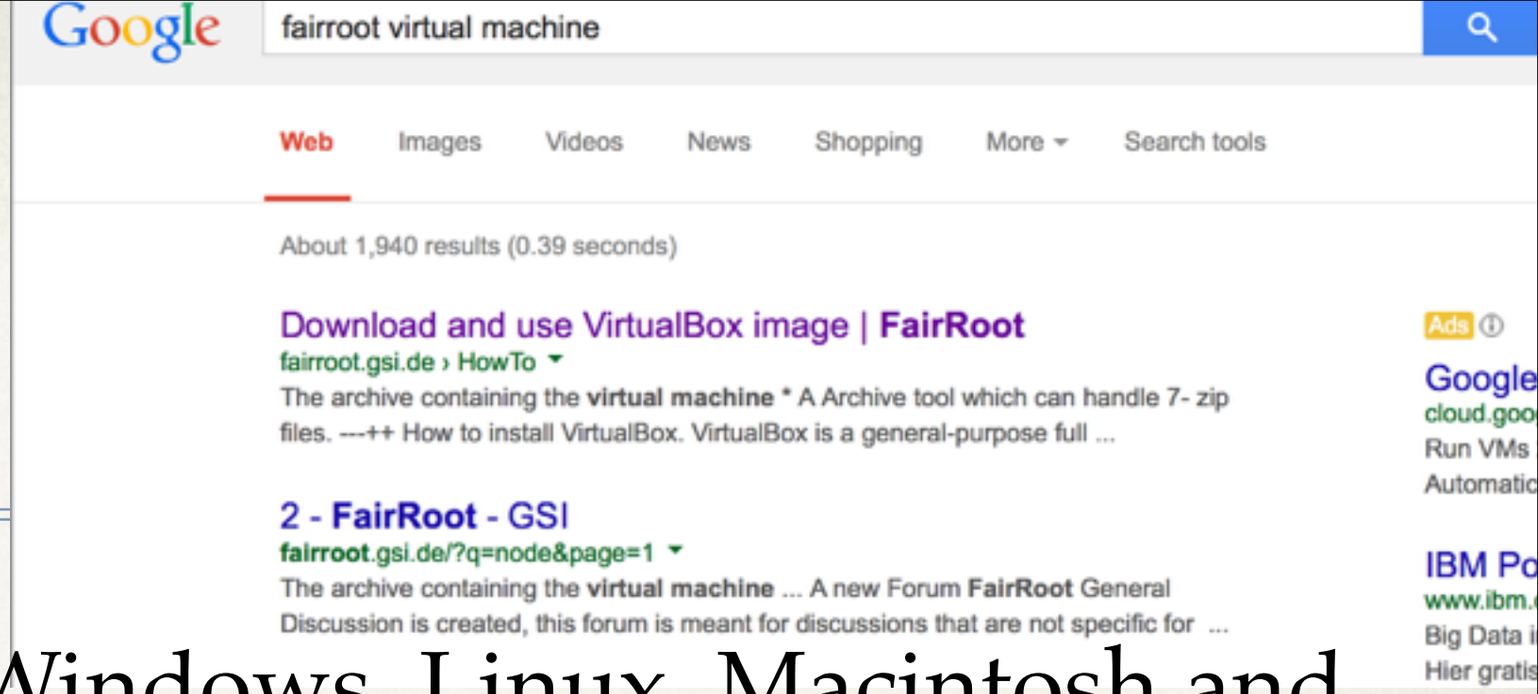
License

FairRoot is distributed under the terms of the GNU Lesser General Public Licence version 3 (LGPL) version 3.

Getting started

Please see.: http://fairroot.gsi.de/getting_started for details.

VirtualBox



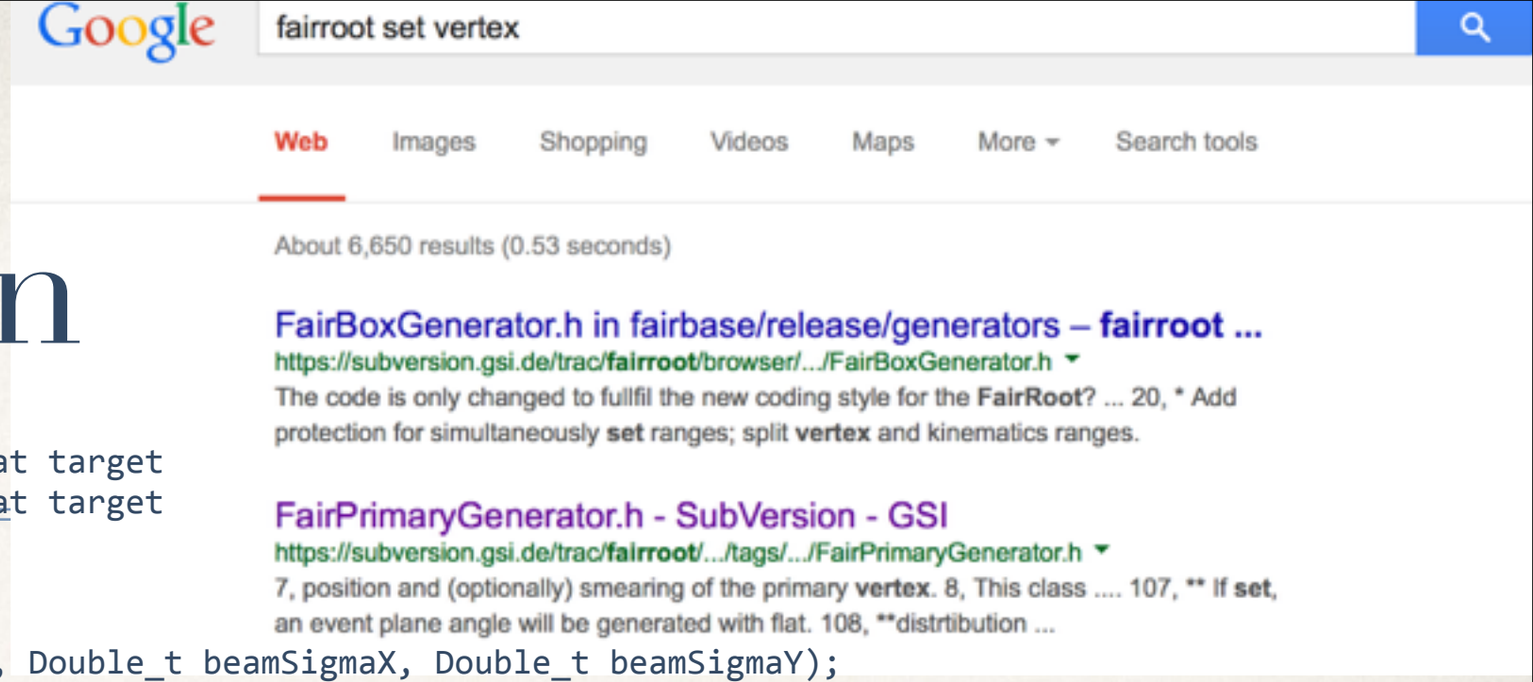
- VirtualBox runs on Windows, Linux, Macintosh and Solaris hosts, while KVM is only available for Linux,

- virtual boxes for pandaroot and cbmroot created in 2012,

- detailed instructions under: http://fairroot.gsi.de/?q=node/74

- no interest from collaborations seen till last Monday.

Vertex position




```
/** Set beam position and widths.
 *@param beamX0      mean x position of beam at target
 *@param beamY0      mean y position of beam at target
 *@param beamSigmaX  Gaussian beam width in x
 *@param beamSigmaY  Gaussian beam width in y
 **/
void SetBeam(Double_t beamX0, Double_t beamY0, Double_t beamSigmaX, Double_t beamSigmaY);

/** Set nominal beam angle and angle widths.
 *@param beamAngleX0      mean x angle of beam at target
 *@param beamAngleY0      mean y angle of beam at target
 *@param beamAngleSigmaX  Gaussian beam angle width in x
 *@param beamAngleSigmaY  Gaussian beam angle width in y
 **/
void SetBeamAngle(Double_t beamAngleX0, Double_t beamAngleY0, Double_t beamAngleSigmaX, Double_t beamAngleSigmaY);

/** Public method SetEventPlane
 **@param phiMin   Lower limit for event plane angle [rad]
 **@param phiMax   Upper limit for event plane angle [rad]
 **If set, an event plane angle will be generated with flat
 **distrtibution between phiMin and phiMax.
 **/
void SetEventPlane(Double_t phiMin, Double_t phiMax);

/** Set target position and thickness.
 *@param targetZ   z position of target center
 *@param targetDz  full target thickness
 **/
void SetTarget(Double_t targetZ, Double_t targetDz);

/** Set target position for multiple tagets. The thickness
 * is the same for all targets.
 *@param nroftargets   number of targets
 *@param *targetZ      z positions of target center
 *@param targetDz      full target thickness
 **/
void SetMultTarget(Int_t nroftargets, Double_t *targetZ, Double_t targetDz);
```

ALFA - common framework for ALICE and FAIR

- Planned online system requirements of ALICE and FAIR experiments:

	PANDA	CBM	ALICE
event rate	20 MHz	10 MHz	50 kHz
event size	4-20 kB	1 MB	20 MB
readout stream	~280 GB/s	~1 TB/s	~1.1 TB/s
storage	~25 kEvents/s	~1 GB/s	20 GB/s
data reduction factor	**~800**	**~1000**	**~60**
remarks	no hardware trigger, event selection in compute nodes	trigger signatures are complex and require partial event reconstruction	reduction of event size necessary, no event selection

Strategies

- Massive data volume reduction by (partial) online reconstruction and compression.
- Much tighter coupling between online and offline reconstruction software.
- ALICE O^2 and FairRoot both aim at sharing online and offline reconstruction code.

Why to collaborate?

- For ALICE experiment: FairRoot already provides some of the needed features (like continuous read-out, building and testing system, FairMQ).

- For FAIR experiments: the common framework will be tested with real data and existing detectors before the start of the FAIR facility on unachievable scale during the design stage.

Who is in the ALFA collaboration?

- FairRoot,
- ALICE Offline,
- ALICE HLT,
- ALICE DAQ,
- CBM,
- FAIR and non-FAIR experiments are welcome to join.

ALFA

- Relies on a data-flow based model (Message Queues).
- Contains:
  - Transport layer,
  - Configuration tools,
  - Management and monitoring tools.
- Provides unified access to configuration parameters and databases.
- Includes support for a heterogeneous and distributed computing system.
- Incorporates common data processing components.

ALFA within FairRoot scheme

AliRoot6 (O^2)

CbmRoot

PandaRoot

R3BRoot

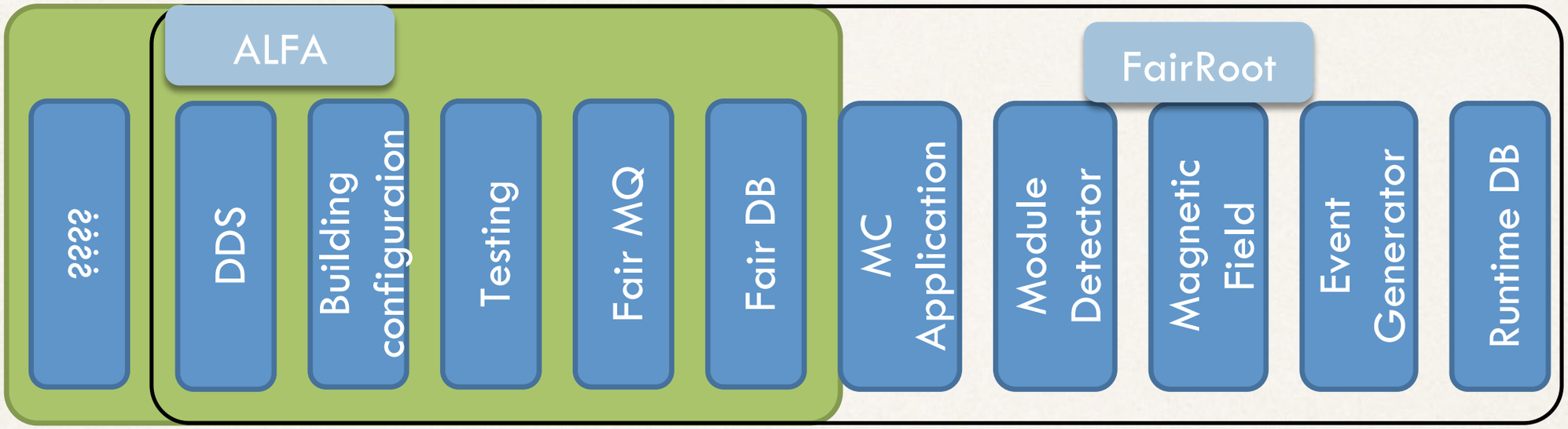
AsyEosRoot

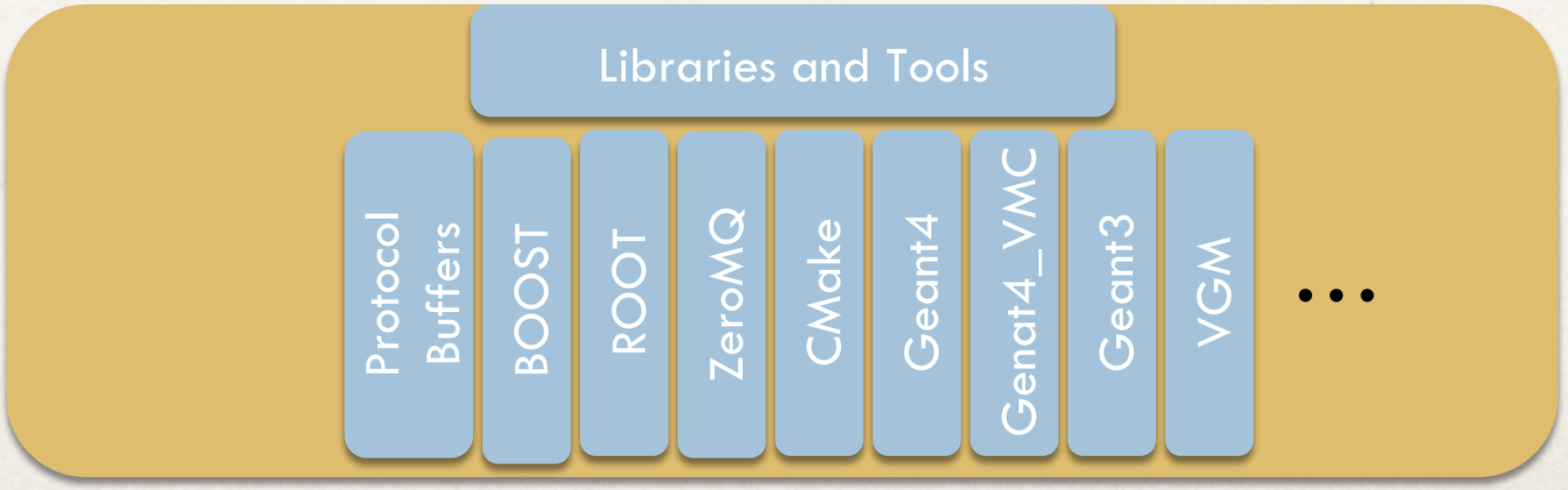
SofiaRoot

FopiRoot

MPDRoot

EICRoot





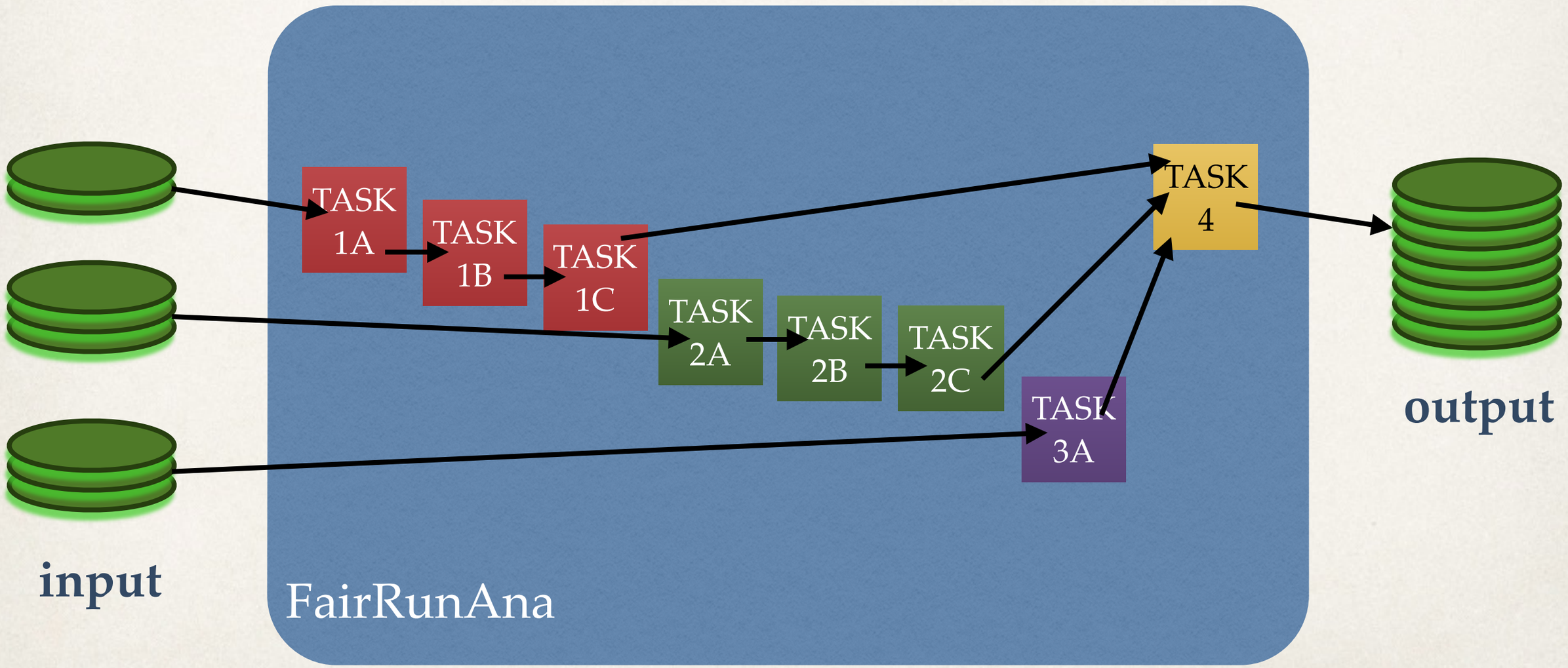
FairMQ

Alexey Rybalchenko, GSI

- FairMQ is an asynchronous messaging toolkit that aims to support online/offline processing and data flow.
- It allows to distribute processes on different nodes and provides the communication layer between these processes.
- Based on socket API.
- Messaging patterns include: publish-subscribe, request-reply, push-pull (pipeline).
- Communication layer is currently based on ZeroMQ or nanoMSG.

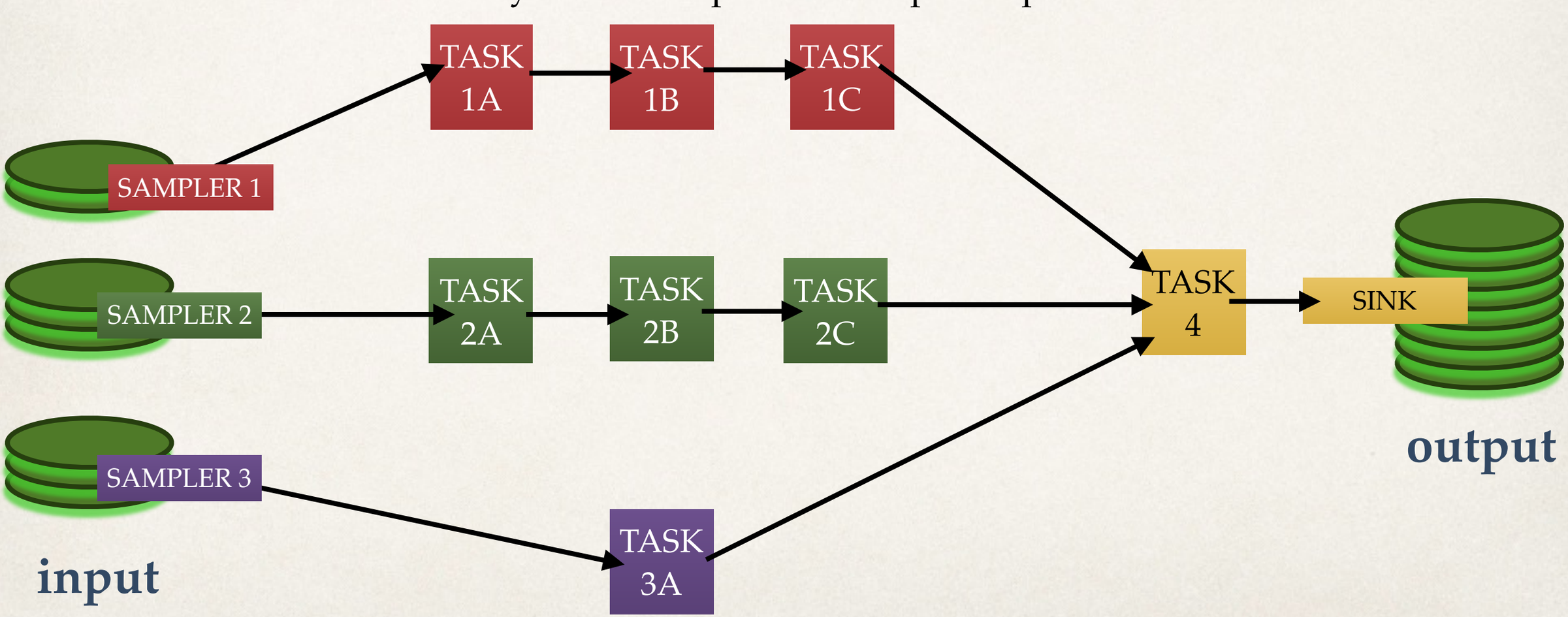
FairRunAna

- Tasks are executed sequentially.

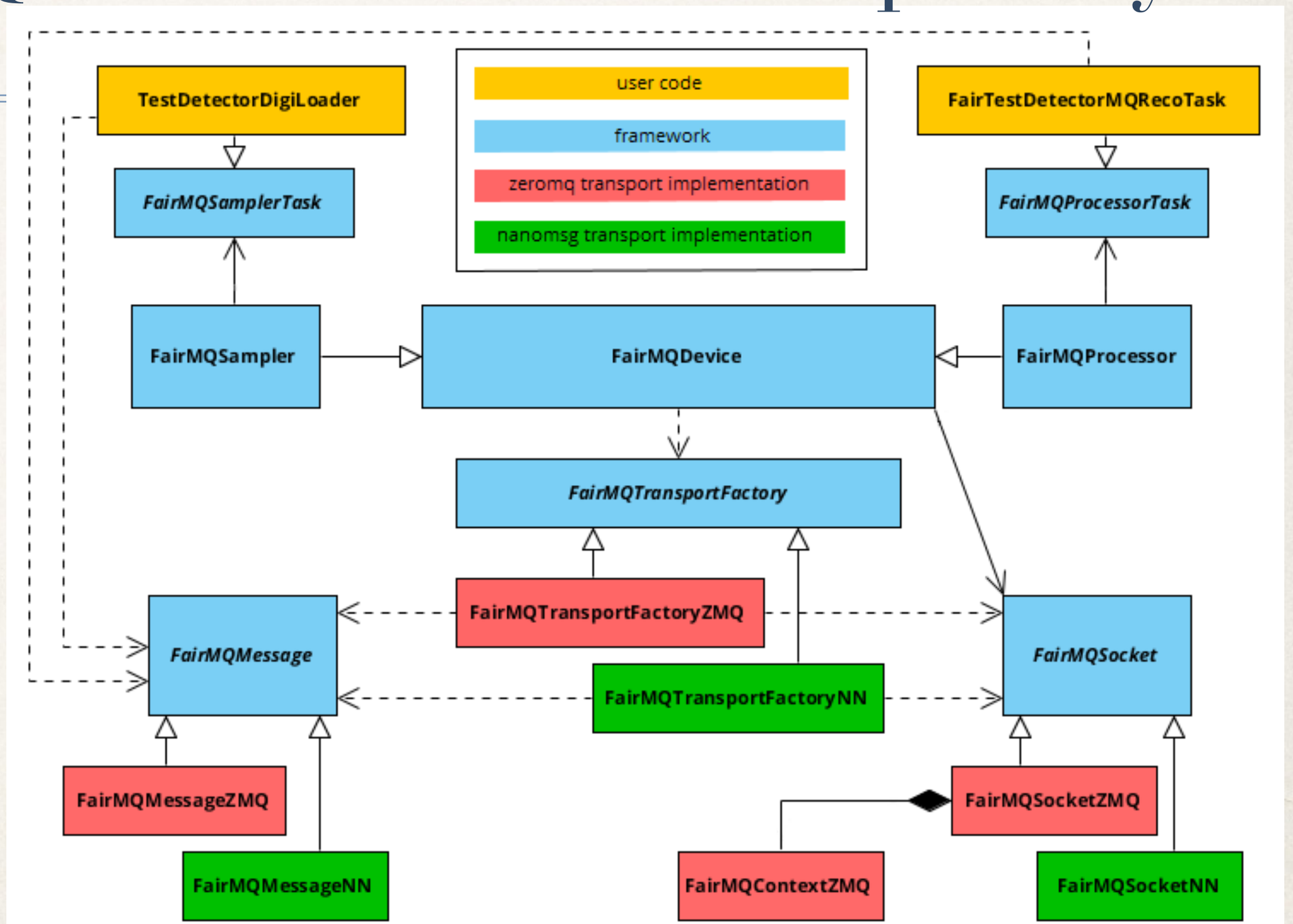


FairMQ

- Each task is a separate process, that could be multithreaded or use GPU.
- Processes may run on a different computing nodes.
- Use RDMA or shared memory whenever possible to speed up communication.



FairMQ has an abstract transport layer



ØMQ

http://zeromq.org

- FairMQ uses ZeroMQ to connect different pieces together:
  - BSD sockets API
  - Bindings for 30+ languages
  - Lockless and fast
  - Automatic reconnection
  - Multiplexed I/O

nanomsg

http://nanomsg.org

- nanomsg is under development by the original author of ZeroMQ
  - 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.
  - **FairRoot is independent from the transport library:**
    - **Modular/Pluggable/Switchable transport libraries.**

Fair_MQ_Device_s

- FairMQSampler - reads and sends the data from the input file;
- FairMQProcessor - wrapper of data processing tasks, it receives, processes and sends data;
- FairMQSink - receives and writes the data to the output file;
- FairMQProxy - allows for splitting or merging data streams.

Data exchange

- Tasks may be running on different hardwares and may be read by different languages - how should the data exchange be done? What about schema evolution?

- Currently following data transports are available:
  - Protocol buffers,
  - Boost serialisation,
  - Binary transport,
  - ROOT TMessages.

Check `example/Tutorial3` for examples

Data exchange

- Google Protocol Buffers, language and platform neutral:
  - "think XML, but smaller, faster, and simpler".

- Boost serialisation, standard C++, versioning:
  - used by CBM Online group.

- binary transport - is not hardware independent:
  - still the fastest, protobuf is 2nd, 3rd - boost.

- ROOT TMessages - ROOT streaming:
  - probably the easiest to implement, yet the slowest.

Data exchange

- Boost:
  - we are generic in the tasks, but intrusive in the data classes (digi, hit, timestamp), need to introduce `serialize` method with data to be boost-serialised.

- binary and protobuf:
  - we are generic in the data class, but intrusive in the tasks (need to fill/access payloads from class with proper setters and getters).

Running FairMQ

Check `example/Tutorial3` for examples

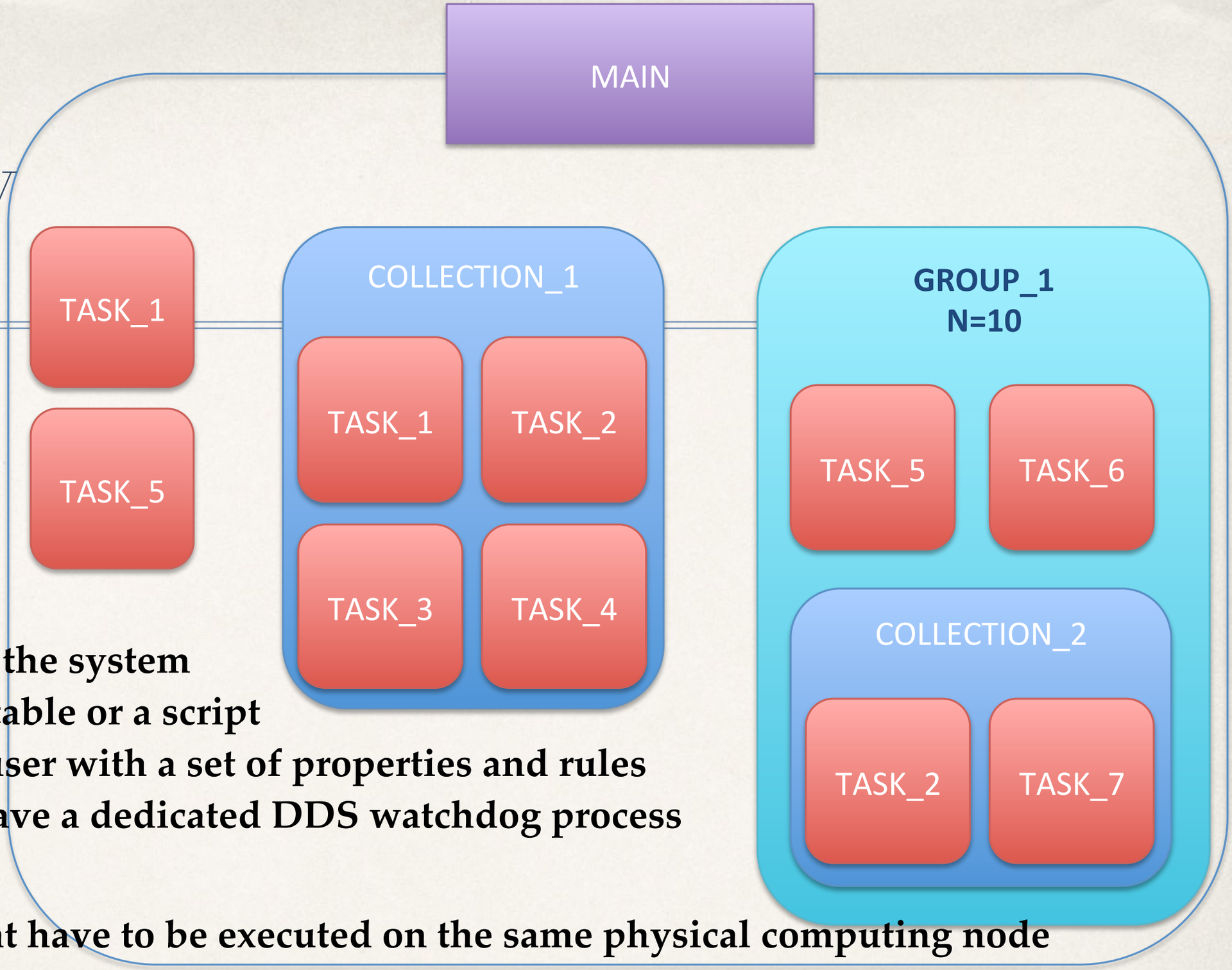
- It is very easy to run the system with a set of shell scripts.
- However deployment of complex topologies would be too complicated and difficult to control.
- Also, we need to utilise any RMS, with an option to run without any RMS, with plain ssh connectivity.
- Furthermore, we should support different topologies and process dependencies.

DDS - Dynamic Distribution System

Anar Manafov, GSI
Andrey Lebedev, GSI

- Available from https://github.com/FairRootGroup/DDS.
- An independent set of utilities and interfaces, which provide a dynamic distribution of different user processes by any given topology on any RMS.
- It should:
  - use (utilize) any RMS (Slurm, Grid Engine, etc),
  - deploy tasks or sets of tasks,
  - provide secure control of nodes (watchdog),
  - support different topologies and task dependencies,
  - support a central log engine.
- Web-based interface allowing user-friendly control of the running system is under development.

Topology



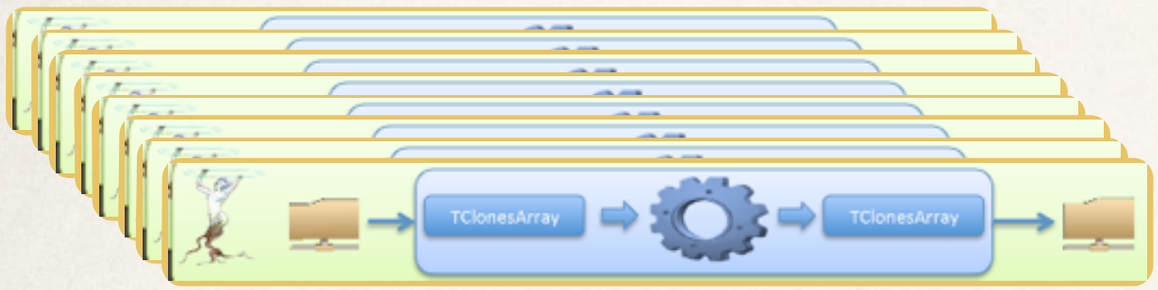
- **Task:**
  - **a basic entity of the system**
  - **can be an executable or a script**
  - **is defined by a user with a set of properties and rules**
  - **each task will have a dedicated DDS watchdog process**
- **Collection:**
  - **a set of tasks that have to be executed on the same physical computing node**
- **Group:**
  - **a container for tasks and collections**
  - **only Main group may contain other groups**
  - **only group define multiplication factor for all its daughter elements**

Simple topology - results

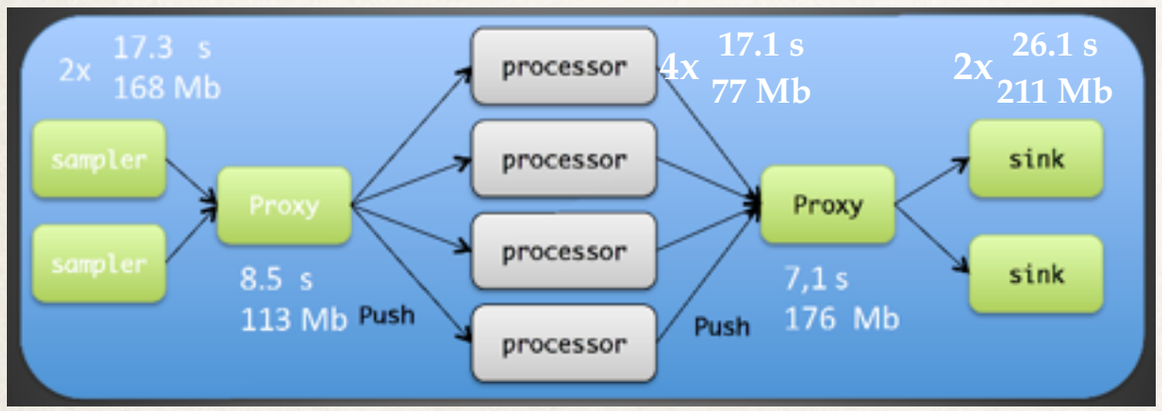
- hit finding in a simple silicon detector on a 2x2.4 Xeon Quad, 16 GB Memory

263 Mb

8x263 Mb

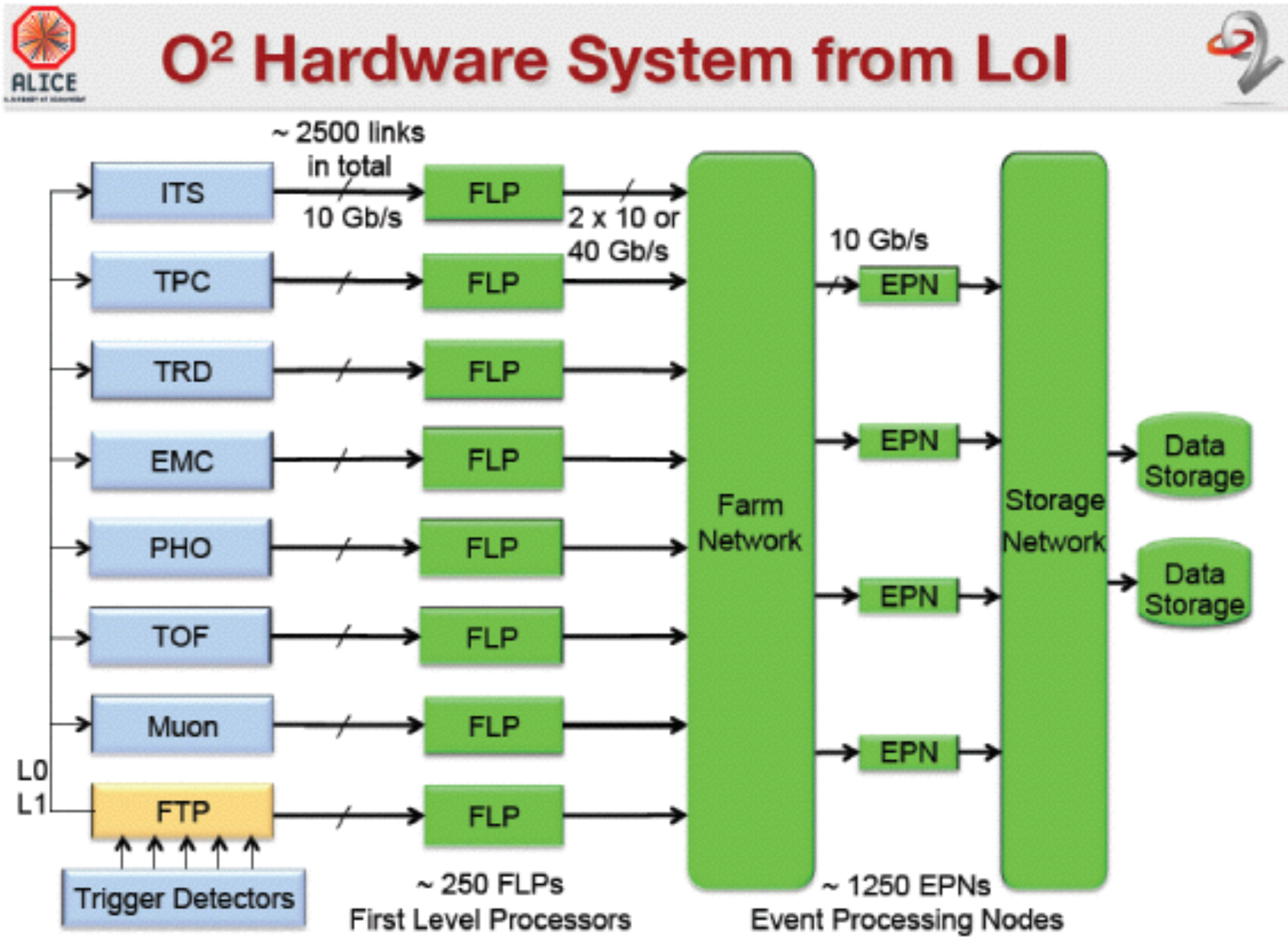


1355 Mb

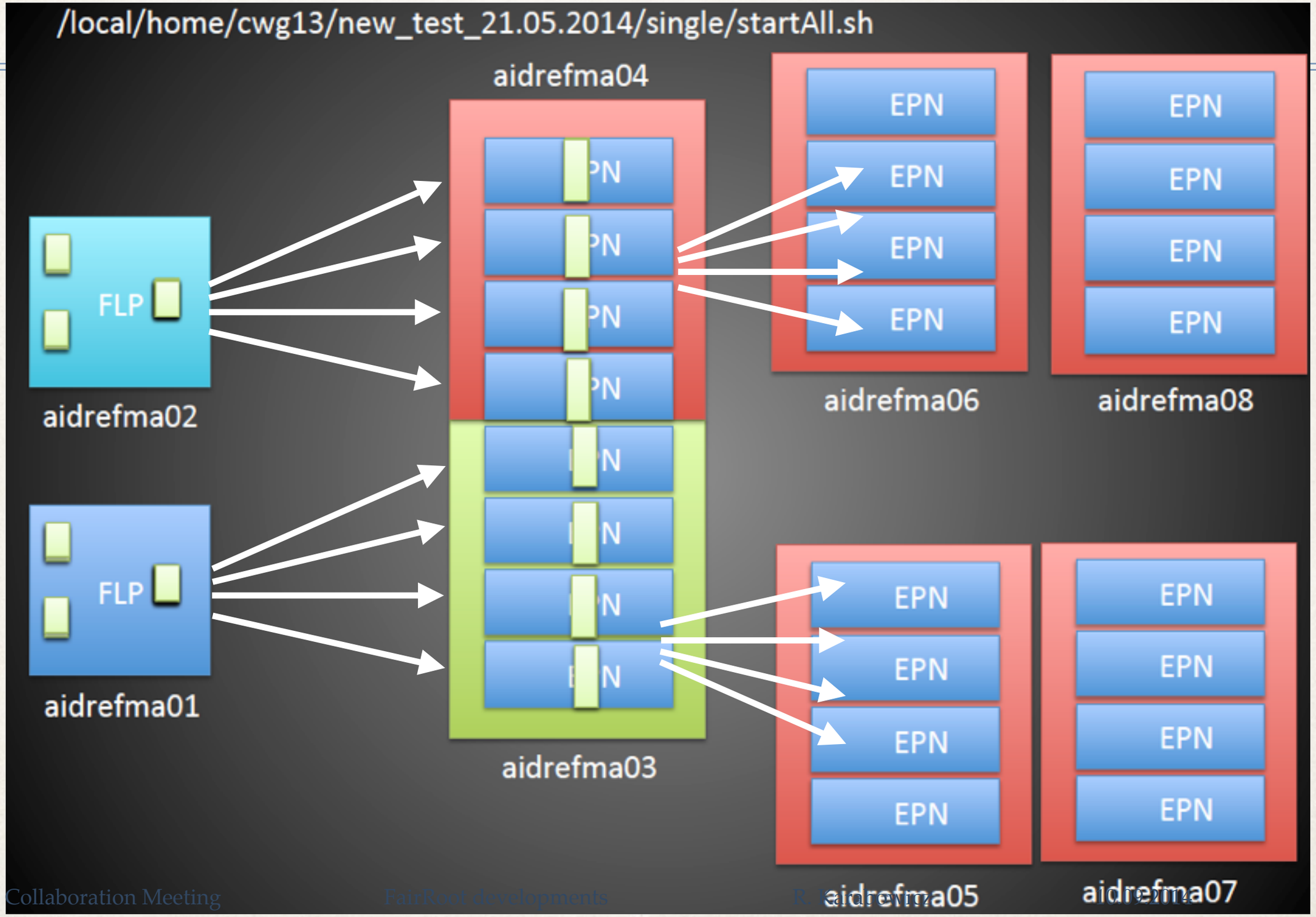


100 events / s
266 events / s
766 events / s

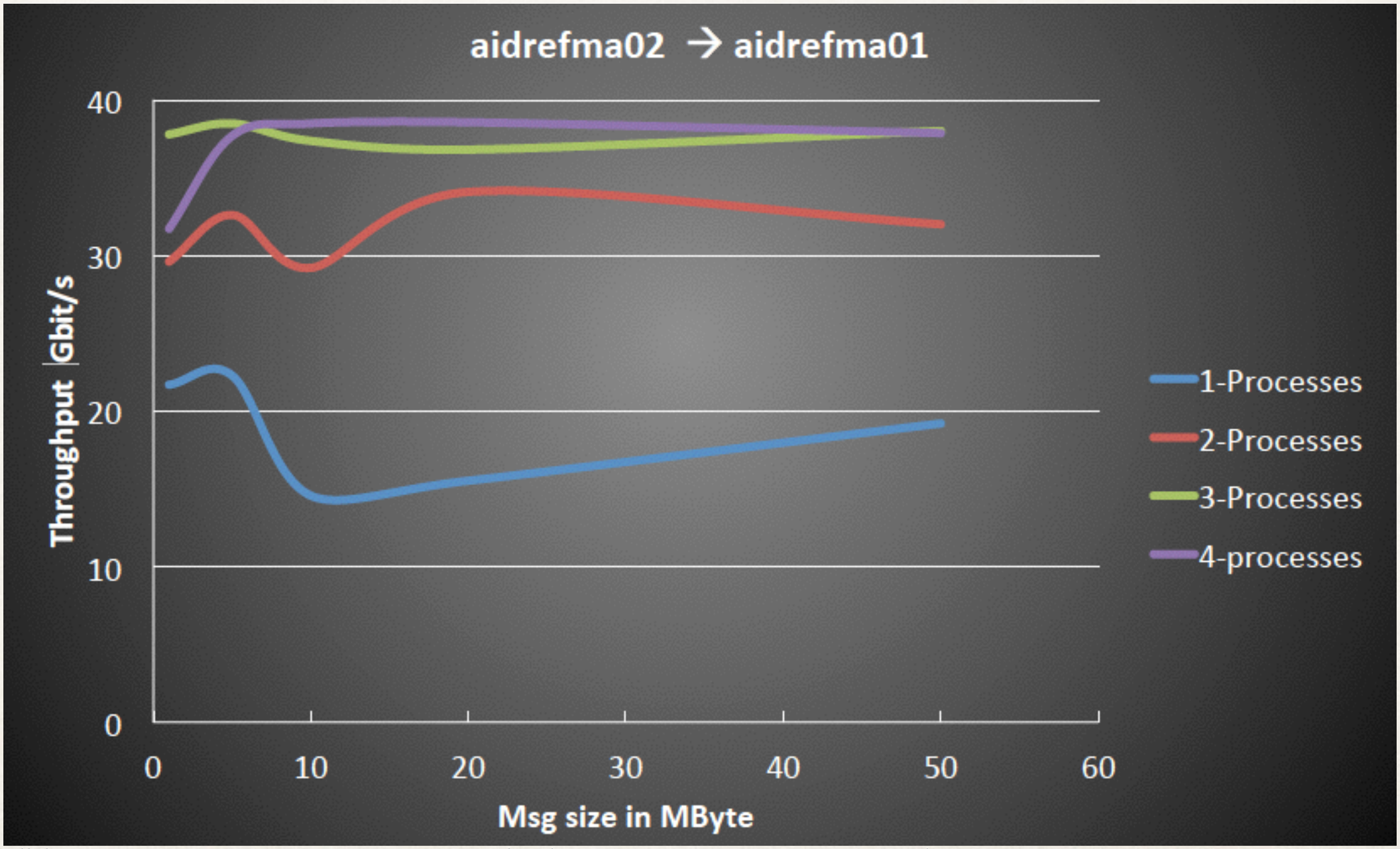
ALICE O^2 DAQ


ALICE
O² Hardware System from LoI
~ 2500 links
in total
10 Gb/s
2 x 10 or
40 Gb/s
10 Gb/s
ITS
TPC
TRD
EMC
PHO
TOF
Muon
FTP
FLP
Farm
Network
EPN
Storage
Network
Data
Storage
L0
L1
Trigger Detectors
~ 250 FLPs
First Level Processors
~ 1250 EPNs
Event Processing Nodes

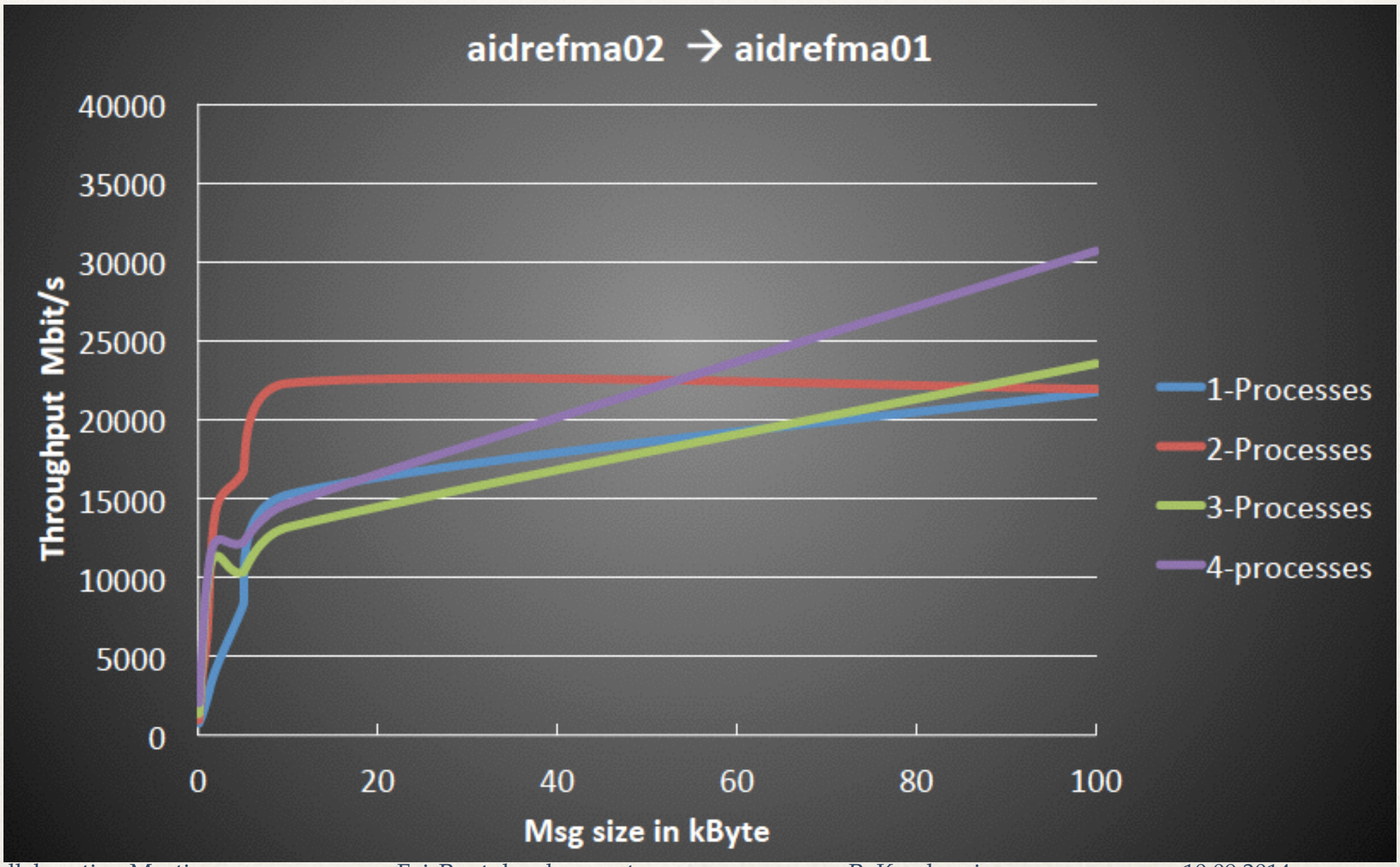
Prototype FairMQ Topology



Running the ZeroMQ performance test on the DAQ test cluster

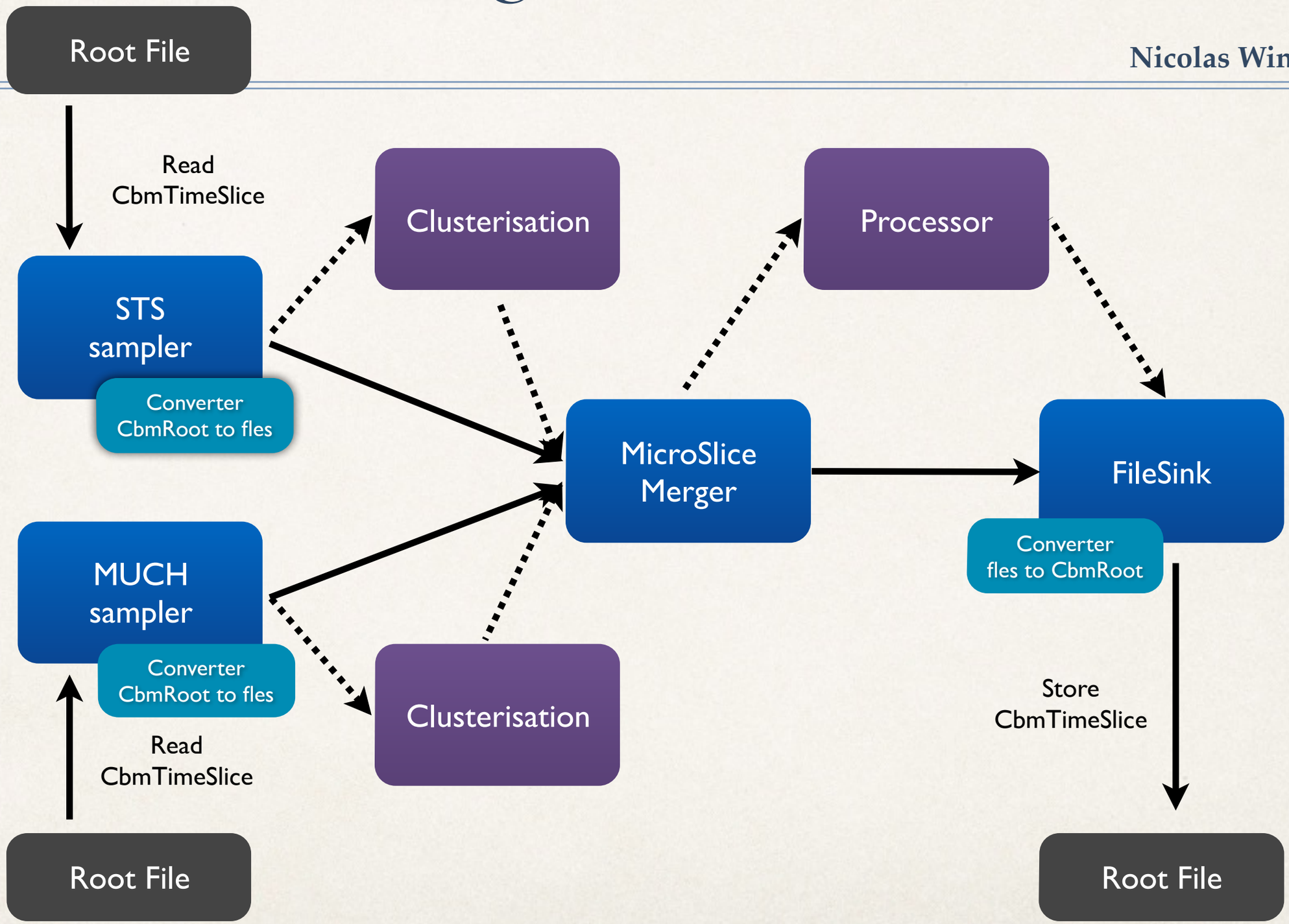


Running the ZeroMQ performance test on the DAQ test cluster



Microslice merger for CBM

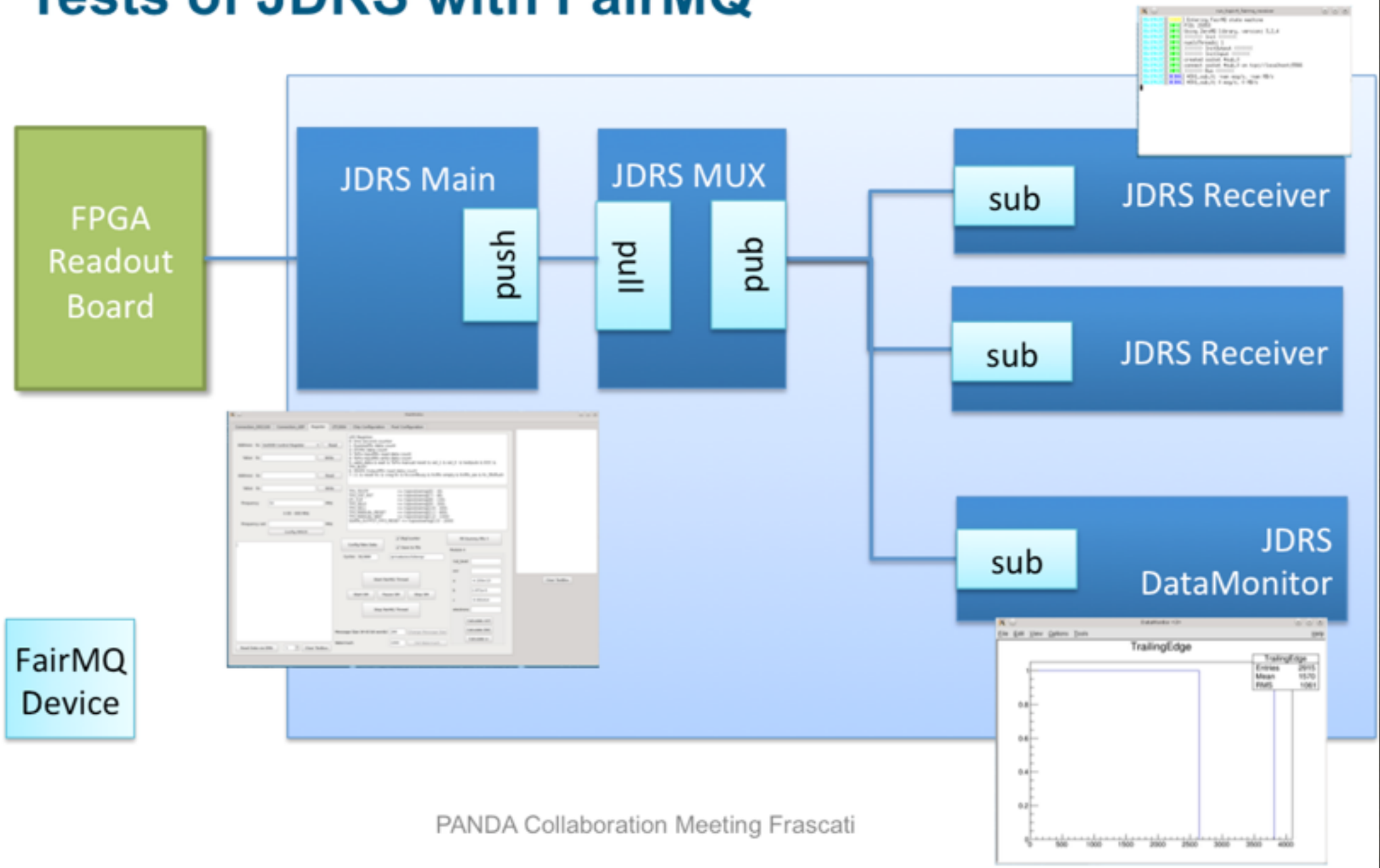
Nicolas Winckler, GSI



Jülich Digital Readout System

Simone Esch

Tests of JDRS with FairMQ



Summary

- newest FairSoft and FairRoot accessible only from github.
- ALFA - project code-name for the set of common tools used by AliRoot and FairRoot.
- MQ - data transport layer is separated from the framework and is currently performed by either ZeroMQ or nanomsg.
- DDS - multipurpose deployment system, avail. Sep 2014.
- ALICE, CBM, PANDA - test systems running.

Special thanks to the FairRoot group

- Mohammad Al-Turany
- Denis Bertini
- Thorsten Kollegger
- Dmytro Kresan
- Andrey Lebedev
- Anar Manafov
- Alexey Rybalchenko
- Florian Uhlig
- Nicolas Winckler