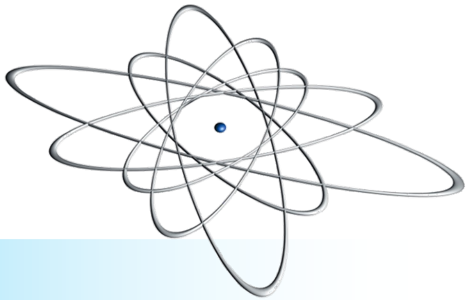
ALFA: A framework for building distributed applications

Mohammad Al-Turany

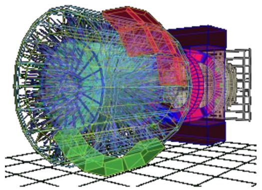
GSI -IT

Research Areas at GSI

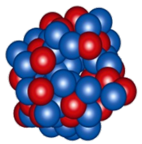


Nuclear Physics

Nuclear reactions
Superheavy elements
Hot dense nuclear matter



Alice



Atomic Physics

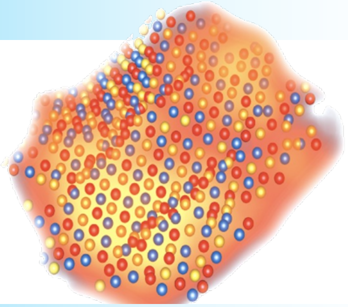
Atomic Reactions
Precision spectroscopy of highly charged ions

Biophysics and radiation medicine

Radiobiological effect of ions
Cancer therapy with ion beams

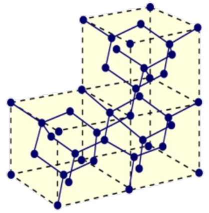
Plasma Physics

Hot dense plasma
Ion-plasma-interaction



Materials Research

Ion-Solid-Interactions
Structuring of materials with ion beams

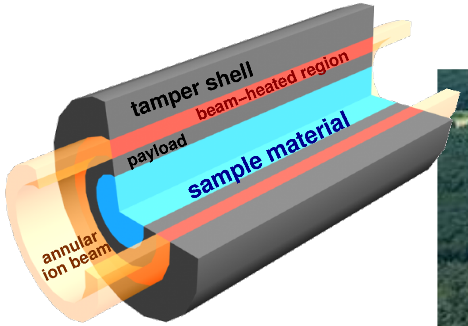


Accelerator Technology

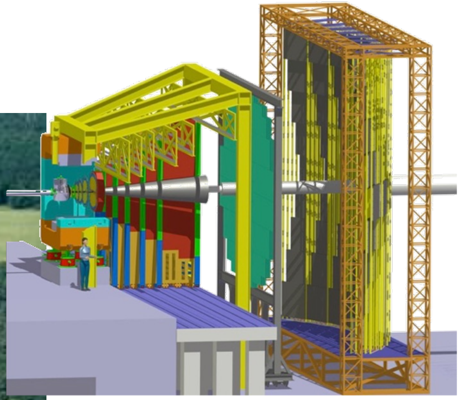
Linear accelerator
Synchrotrons and storage rings

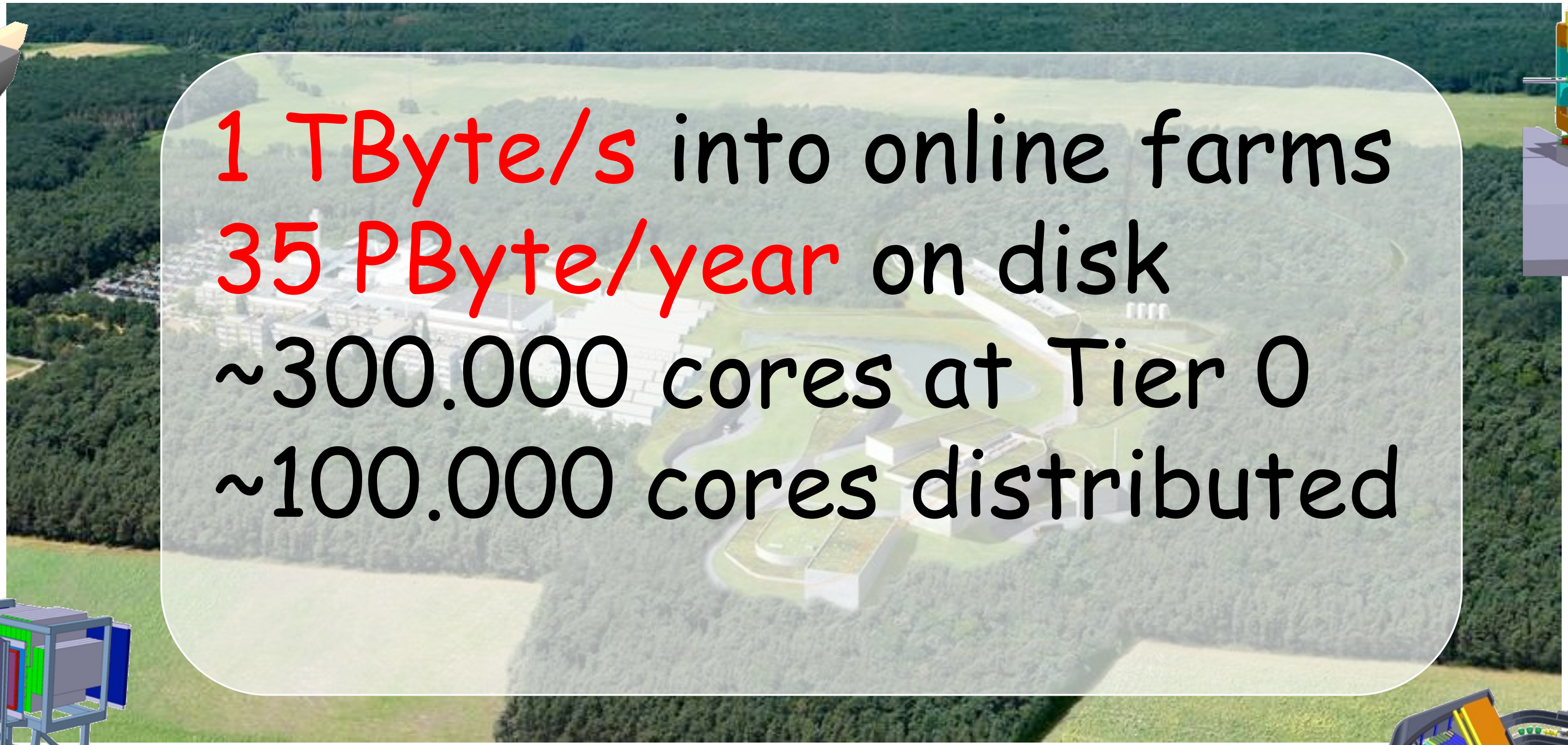
FAIR - four research pillars

APPA

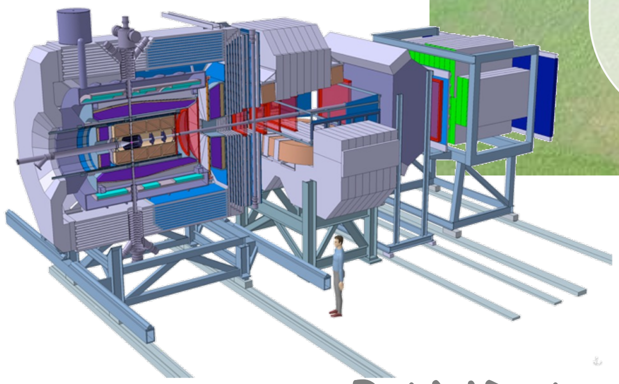


CBM

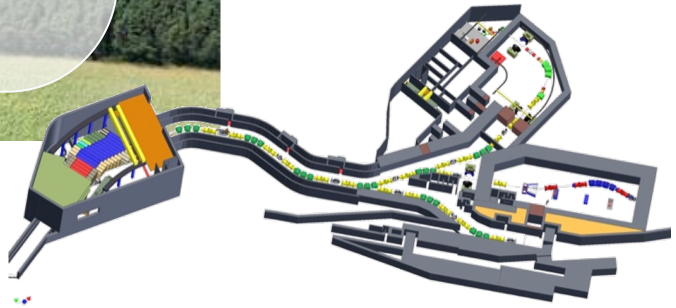




1 TByte/s into online farms
35 PByte/year on disk
~300.000 cores at Tier 0
~100.000 cores distributed



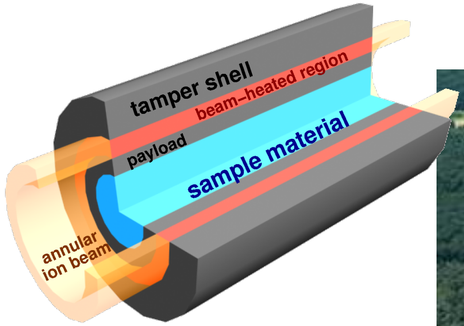
PANDA



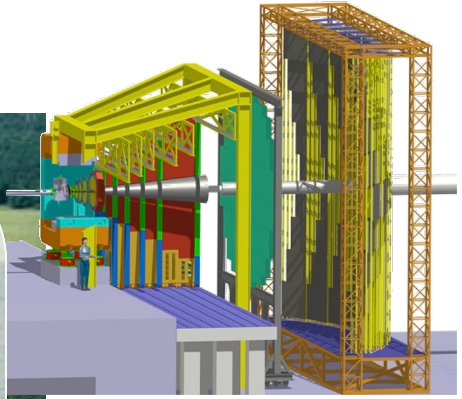
NUSTAR

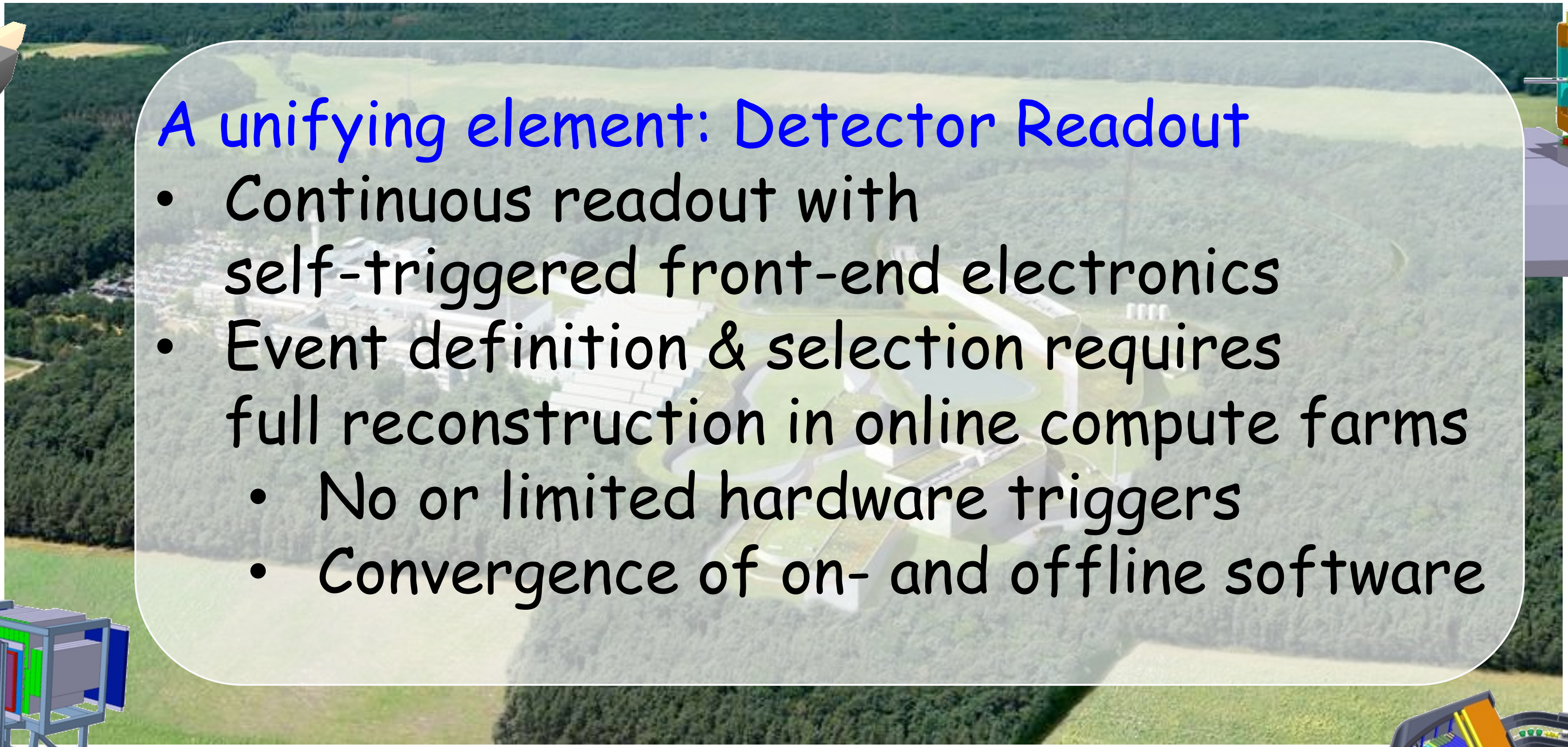
FAIR – four research pillars

APPA



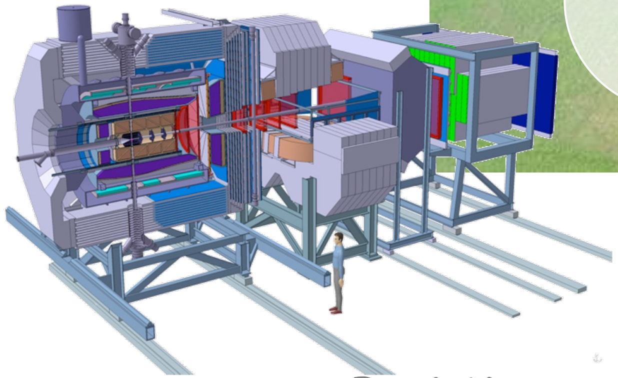
CBM

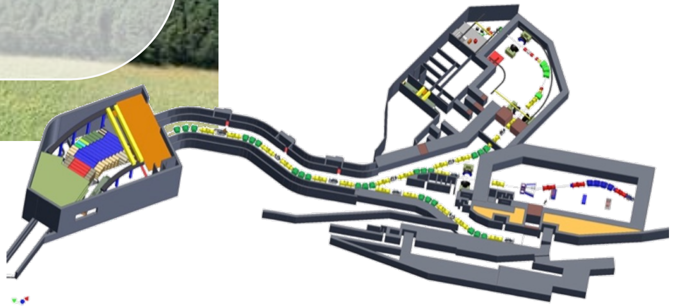




A unifying element: Detector Readout

- Continuous readout with self-triggered front-end electronics
- Event definition & selection requires full reconstruction in online compute farms
  - No or limited hardware triggers
  - Convergence of on- and offline software





NUSTAR

PANDA

2003 : CBM Collaboration

Geant3

AliRoot

FLUKA

PAW

VMC

Geant4

Pythia

ROOT

Urqmd

Go4

Hydra

- We need simulations for the LOI
- We have no manpower for software
- Re-use existing software
- It has to be easy, fast, reliable, ..etc
- We need it yesterday

How to allow Physicists to:

- Focus on detector performance details,
- Avoiding purely software-related problems such as storage, retrieval, code organization, etc.;
- Avoid delving into low-level details.
- Use ready-made and well-tested code for common tasks.

How to allow Physici

- Focus on detector r
- Avoiding pur
  storage
- A
- U   ue for common tasks.

Software framework dedicated to particle/nuclear physics simulation and reconstruction based on the experience we gain from working on LHC and GSI experiments

Software for FAIR Experiments (FairRoot)

FAIR and non-FAIR experiments join the effort to build one platform for simulation and reconstruction software

FairRoot

- **AliceO2** http://alice-o2.web.cern.ch/
- **CbmRoot** https://fair-center.eu/for-users/experiments/cbm.html
- **PandaRoot** https://panda.gsi.de/
- **R3BRoot** https://www.gsi.de/r3b
- **FairShip** http://ship.web.cern.ch/ship/
- **SofiaRoot**
- **AsyEosRoot**
- **MPDRoot** http://mpd.jinr.ru
- **ExpertRoot** http://er.jinr.ru/
- **EnsarRoot** http://igfae.usc.es/satnurse/ensarroot.html
- **ATTPCRootv2** https://github.com/ATTPC/ATTPCROOTv2
- **BNMRoot** http://mpd.jinr.ru

FairRoot

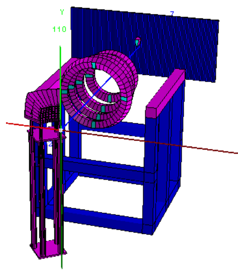
- …
- Testing
- Parameter/ Condition Management
- Geometry
- Detector Response
- Magnetic Field
- Event Generators

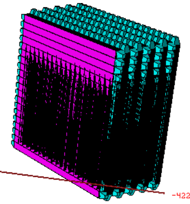
Software Stack

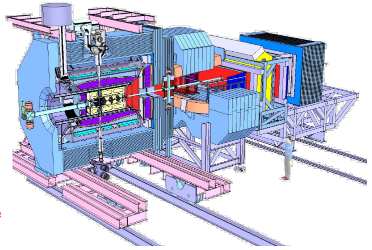
- ROOT
- Boost
- Simulation Engines
- VGM
- CMake
- Network (ZeroMQ, nanomsg, OFI libfabric)
- Serialization (Protocol Buffers, msgpack, FlatBuffers, ROOT)
- Google Test
- …

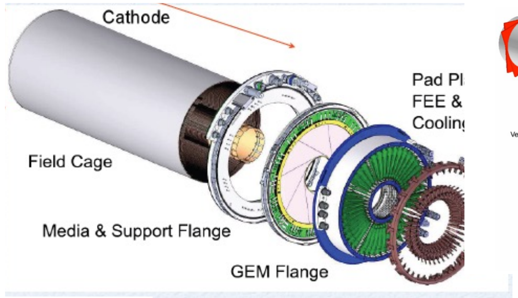
FairRoot a success

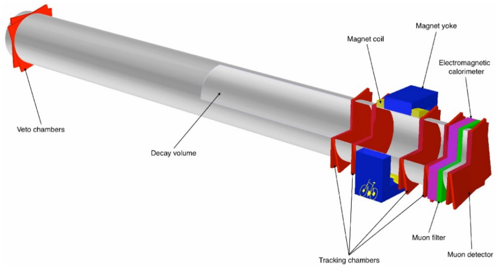
- Used for simulations and design studies for FAIR and Non-FAIR experiments
- It enhanced the synergy between the different groups
- Many useful tools where developed within FairRoot

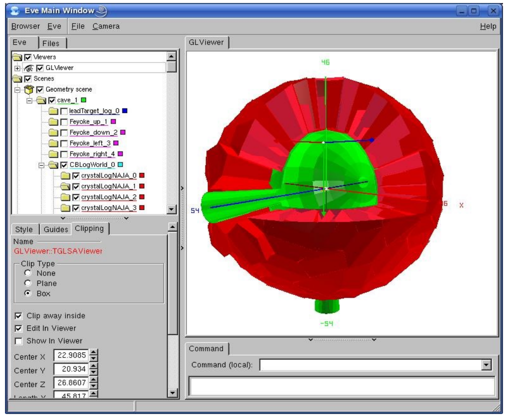


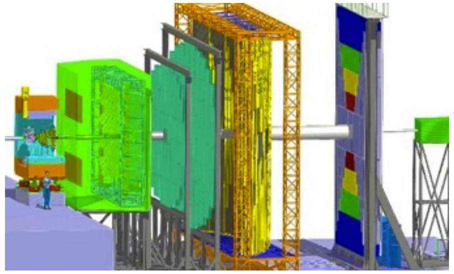


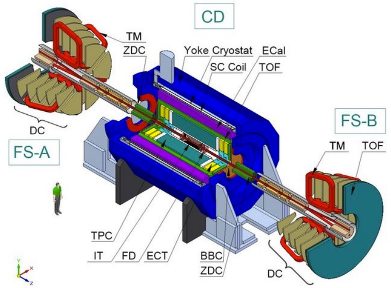


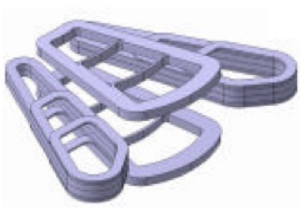


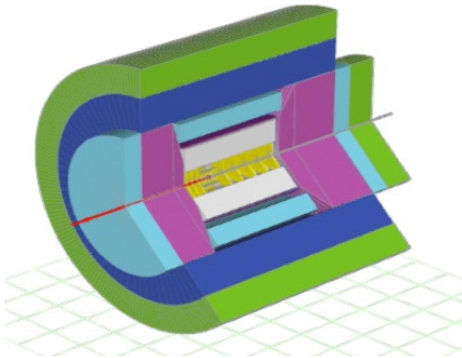












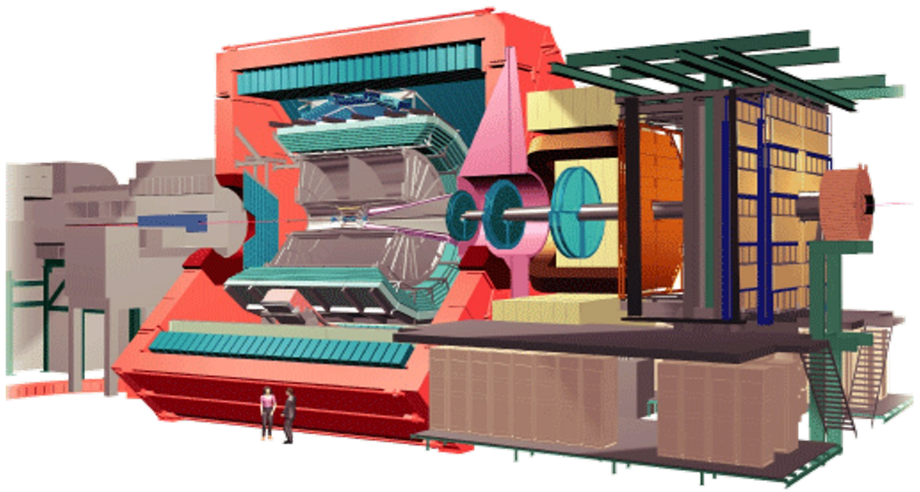
What about..

- Online computing?
  - Handling 1 TByte/s data transport in the online systems
- Support heterogeneous architectures
  - Accelerator cards (GPUs, Xeon Phi)
- Concurrency?
  - Multi-/Many-Core
  - SIMD

ALICE Upgrade

- The Inner Tracking System (ITS) will be replaced with a new, high-resolution, low-material detector
- The Time Projection Chamber (TPC) will be upgraded with replacement of the chambers by Gas Electron Multipliers (GEMs) and a new pipelined readout electronics based on a continuous read-out scheme
- The forward trigger detectors and the electronics of the Transition Radiation Detector (TRD), the Time Of Flight (TOF), and several other detectors will be upgraded

ALICE Upgrade



- continuous readout
- x50 event rate

3.4 TB/s

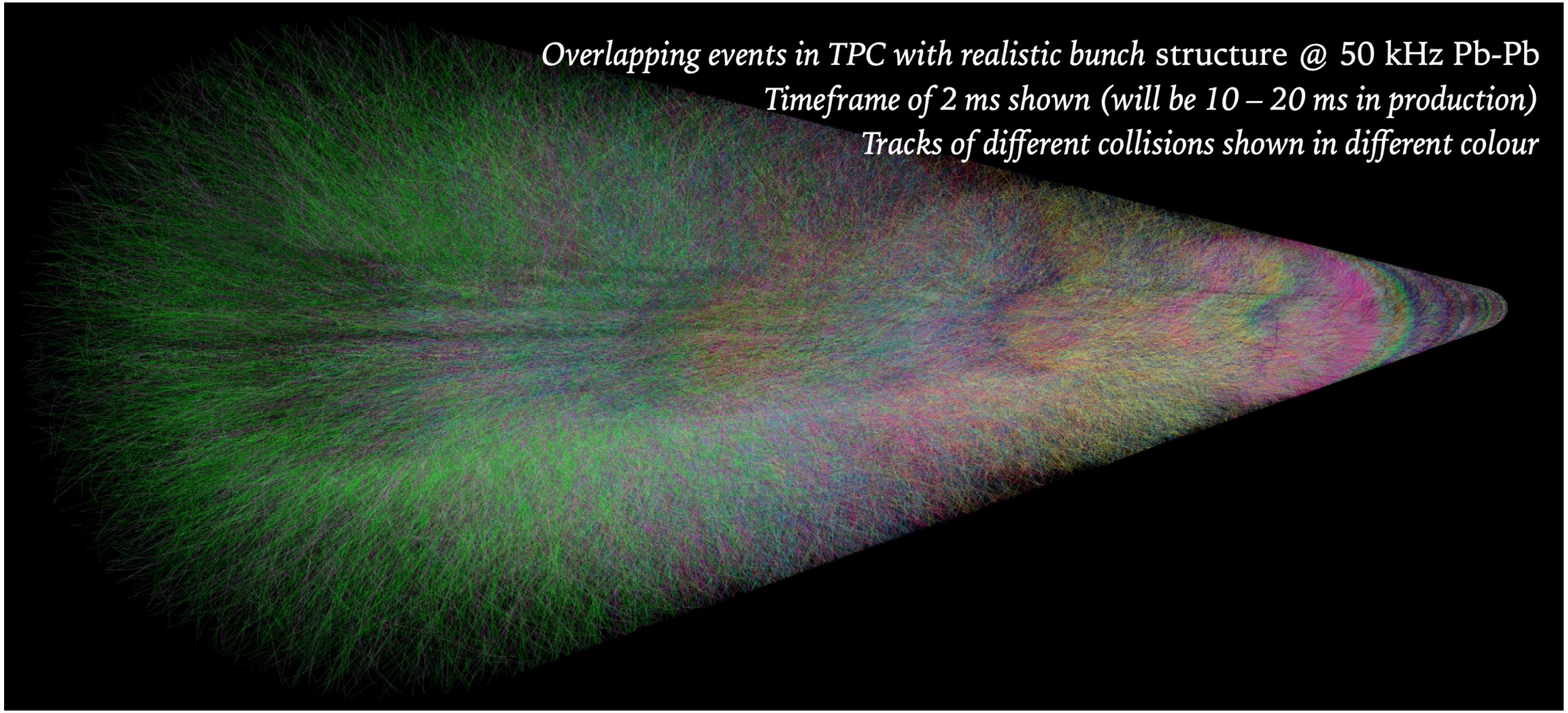
50 kHz

Online/Offline Facility

(50 PB/y) **Storage** 90 GB/s

- Aim is to reduce data volume by doing (quasi) online reconstruction
  - Each and every event needs to be processed, no rejection
- High Throughput (and not Performance) Computing problem

Alice in RUN3
50 kHz of continuous readout data.
90 Gbytes/s to Storage (50 PB/y)



Compared to RUN2

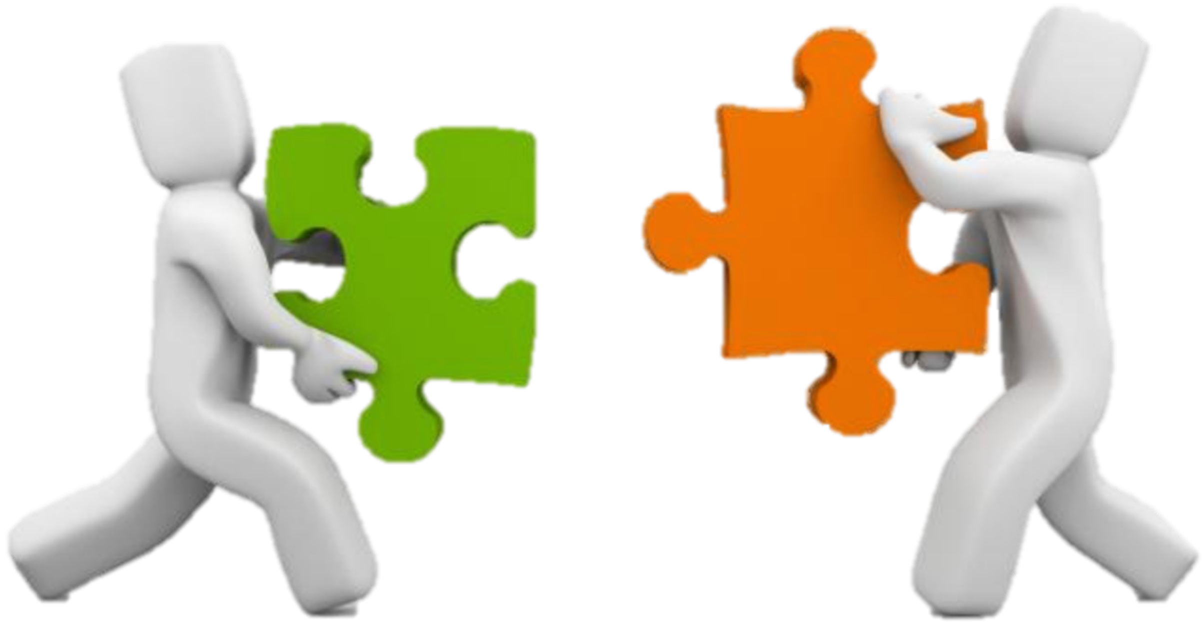
- ***Reconstruct 50x more*** *events online*
- ***Store 50x*** *more events*
- ***continuous readout (****TPC data* ***)*** *in combination with data coming from triggered detectors.*

Two projects – same requirements

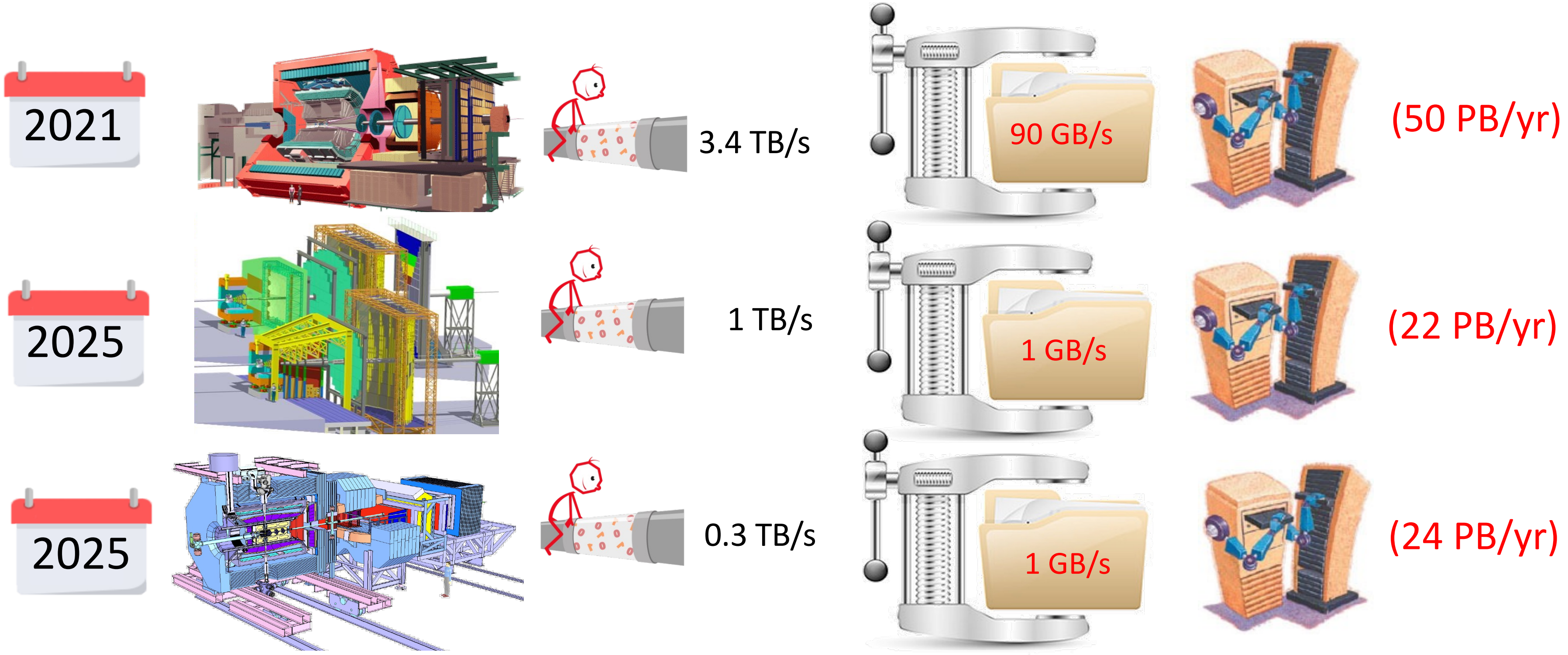
Massive data volume reduction

Data reduction by (partial) online reconstruction

Online reconstruction and event selection



- Reduce data volumes by doing (quasi) online reconstruction
  - Each and every event needs to be processed, no rejection
- High Throughput (and not Performance) Computing problem

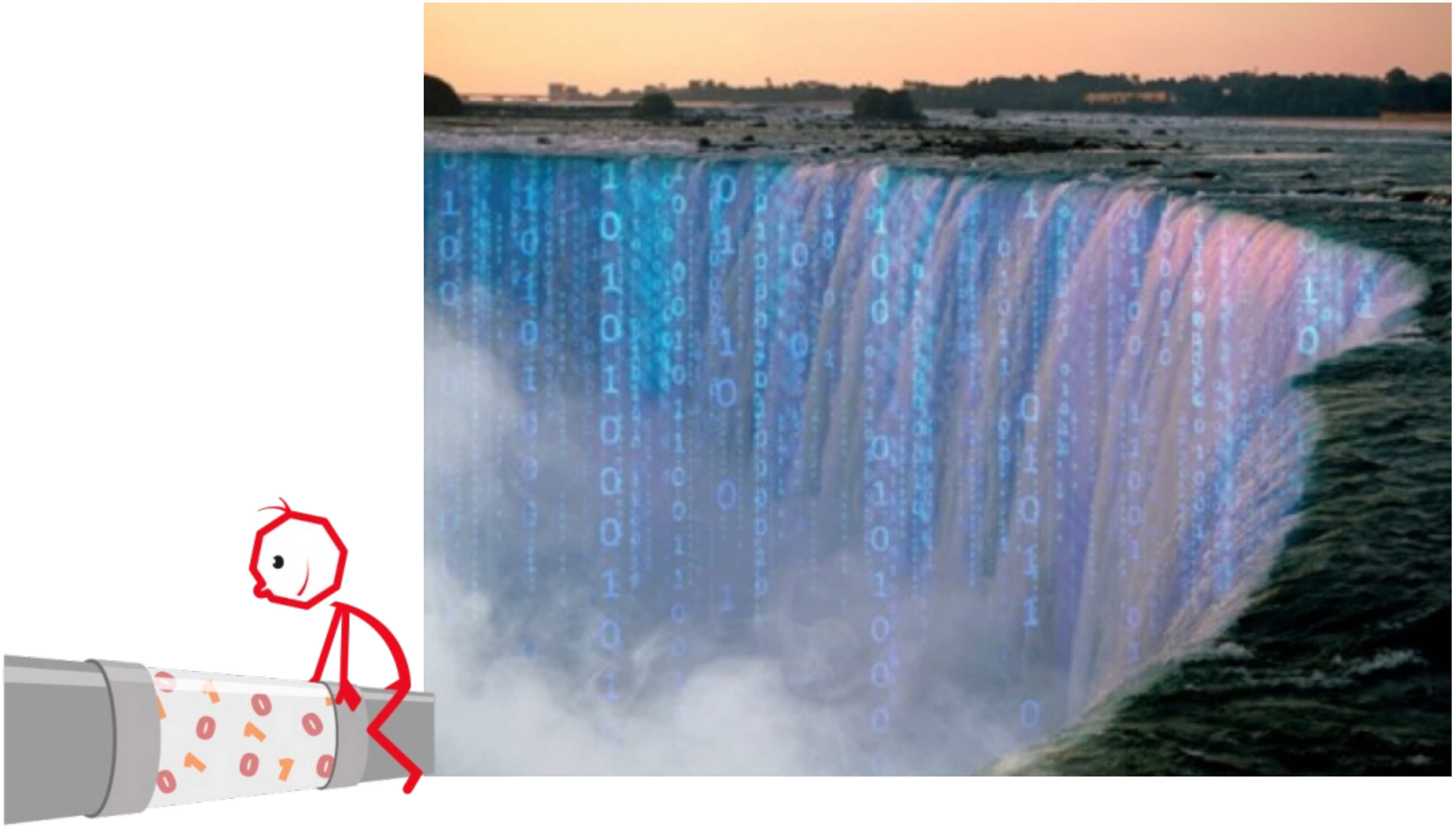


What to do?

- ALICE can cope with the challenges of Run3 only by a radical redesign of its software and computing architecture.
- FAIR Experiments needs a new concept!

What is new in ALFA compared to FairRoot and/or AliRoot?

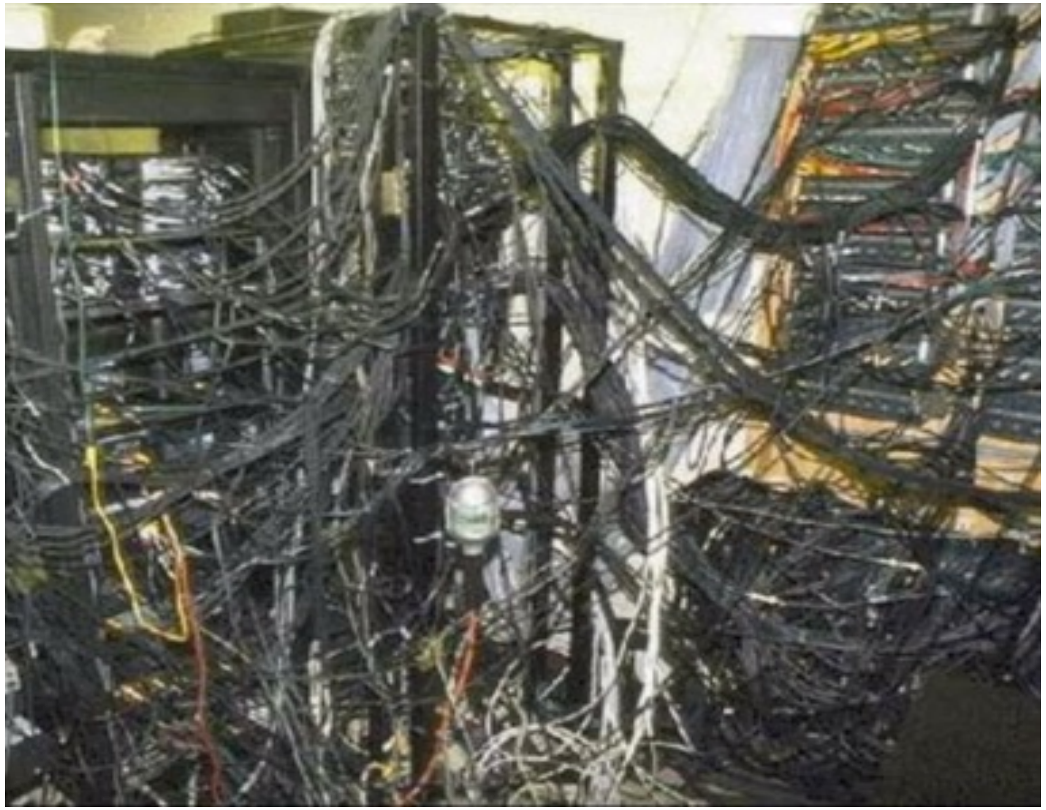
ALFA has a data-flow based model:



Message Queues based multi-processing

Works locally and across most networks!

- Ethernet
  - ZMQ, nanomsg
- InfiniBand (IPoverIB, RDMA)
  - ZMQ, nanomsg, OFI
- Shared Memory Transport
  - Boost



Actor Model

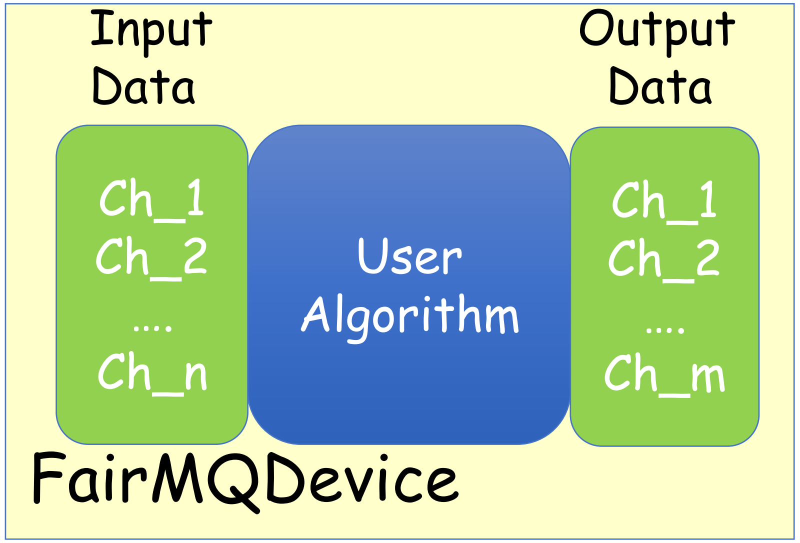
Standalone processes ("devices") perform a task (e.g. track finding) and communicate with each other via messages (mediated by a queue).

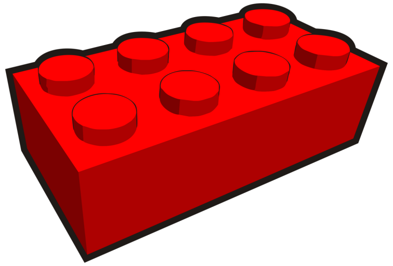
Actor Model

- No locking, each process runs with full speed
- Easier to scale horizontally to meet computing and throughput demands (start/add new instances)

ALFA building block (FairMQ Devices)

- Device takes/passes ownership of data
- Framework user sees only the callback to his algorithm
- Different channels can use different transport engines





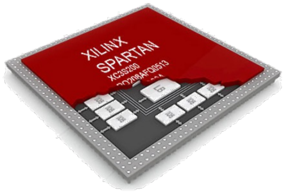
Right tool for the right job!

Each "Task" is a separate process, which:

- Can be multithreaded, SIMDized, …etc.
- Runs on different hardware (CPU, GPU, …, etc.)
- Can be written in any supported language (Bindings for 30+ languages)

Different topologies of tasks can be adapted to the problem itself and the hardware capabilities

Message format ?



The framework does not impose any format on messages.

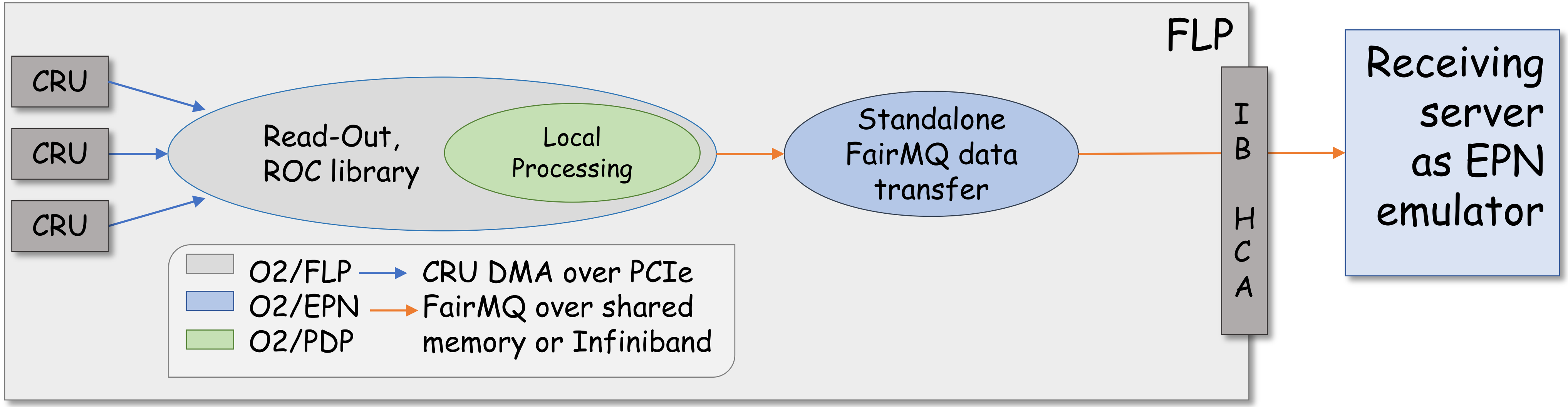
It supports different serialization standards

- BOOST C++ serialization
- Google's protocol buffers
- ROOT
- Flatbuffers
- MessagePack
- User defined

FairMQ Transport: General concepts:

- Hide all transport-specific details from the user.
- Clean, unified interface to different data transports.
- Combinations of different transport in one device in a transparent way.
- Transport switch via configuration only, without modifying device/user code -> same API for all transports.

FairMQ for ReadOut in ALICE

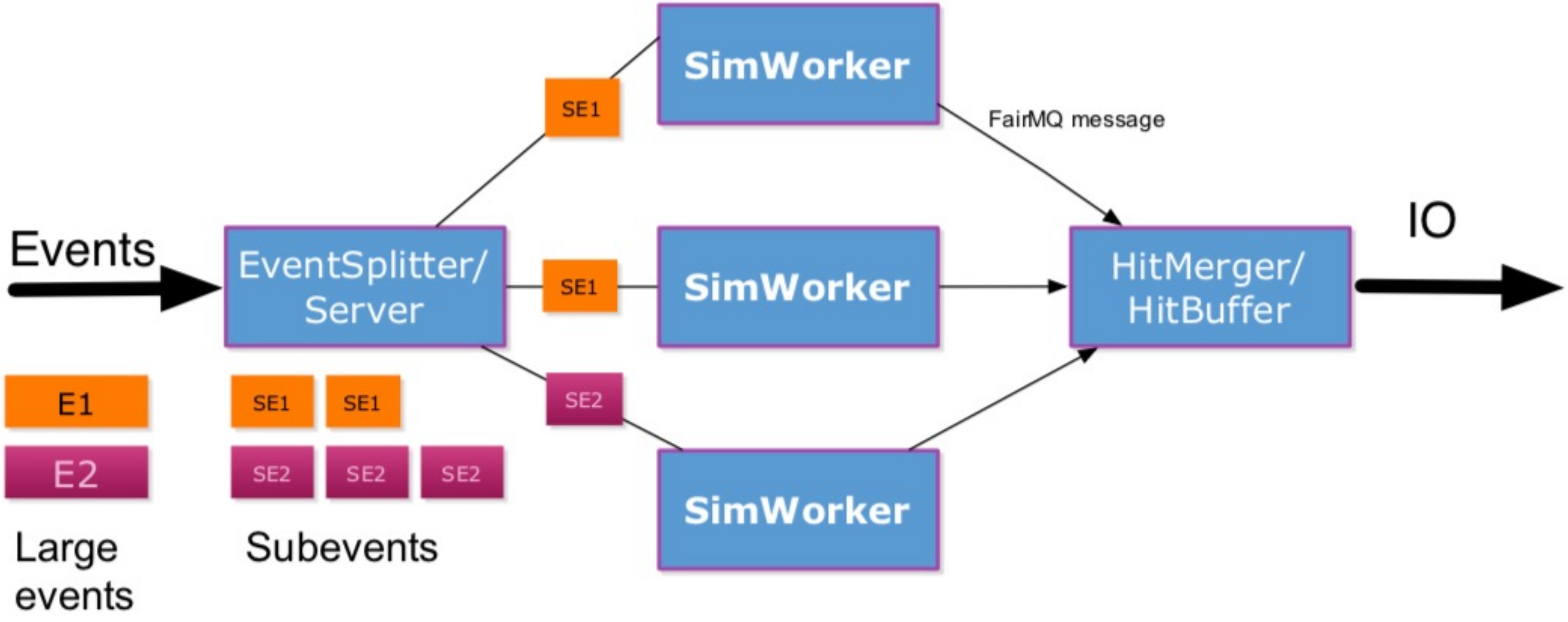


- CRU test data, TPC decoder algorithm integrated in Readout
- Demonstrate usage of available CPU resources at target data throughput

Run chain for 8 hours, use as much CPU as possible at target data throughput

SUCCESS: # CRUs x 17.25 Gb/s with Local Processing active

FairMQ-based parallel simulation



Sandro Wenzel

Analysis in RUN3:

Problem:

Analysis remains I/O bound in spite of attempts to make it more efficient by using the train approach

Analysis in RUN2:

- Organized analysis
- Event-oriented data model: trees of ESD & AOD/delta AOD, but also kinematics, ESD friends, track references, tags
  - Access to the different data via handlers
- Possibility to run in local, Proof, GRID, event mixing modes
  - Services: I/O, event loop, merging of results, bookkeeping
  - LEGO trains
- All user code on GitHub (alisw/AliPhysics) and built centrally on CVMFS

Analysis Trains:

Analysis tasks organized in trains (dependencies, I/O):

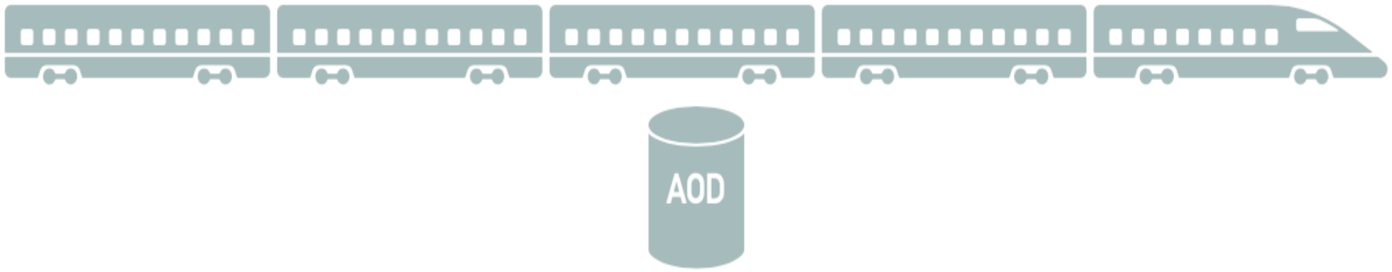
- Read data once,
- process many times,
- benefit from common processing

Compared to RUN2

- ***Reconstruct 50x more*** *events online*
- ***Store 50x*** *more events*
- ***continuous readout (****TPC data* ***)*** *in combination with data coming from triggered detectors.*

Analysis in RUN3: (Solution)

- **Retain concepts that worked:** analysis trains, centralized code, abstraction framework
- Use better compression algorithms
- **Recompute** quantities on the fly rather than storing them.
- Flat data structures
- Only AODs for analysis



Software framework:
O2 Data Model

- ALICE-specific description of the messages between devices
- Computer language agnostic, extensible, efficient mapping of the data objects in shared memory or to the GPU memory
- Supports multiple data formats and serialization methods

Requirements for the AOD format

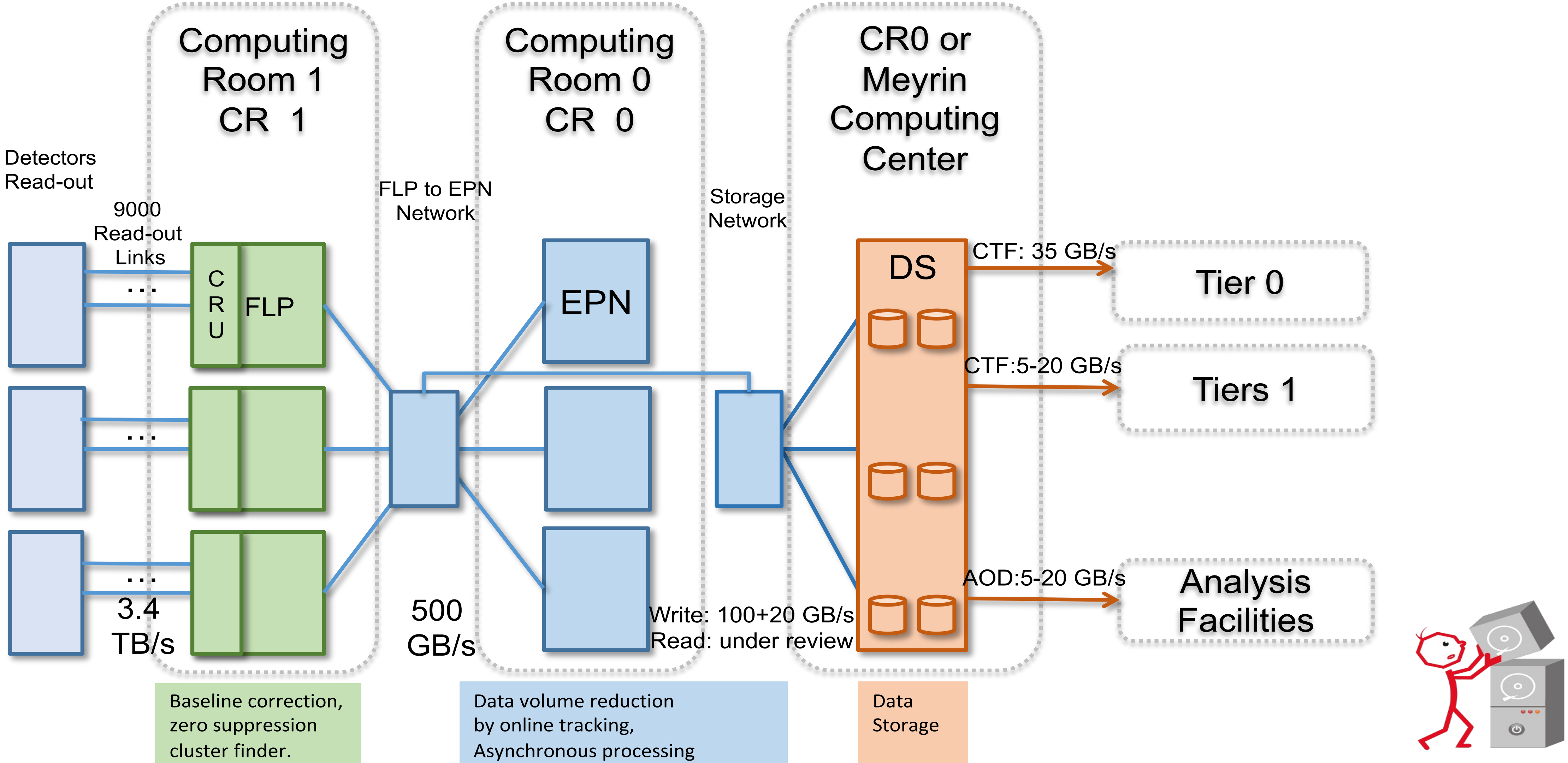
- AOD's data format will have to play well with AliceO2 message passing, shared memory backed, distributed nature.
- **Zero-{Copy, Serialisation, Adjustments}**:
  - *we want to be able to reuse data between processes.*
- **Growable**: *ability to extend columns on the fly.*
- **Prunable:** *ability to drop columns on the fly.*
- **Skimmable:** *ability to select only certain rows.*
- *Strategy: we are willing to lose some degree of generality for performance.*

Apache Arrow

Data Analysis using ALICE Run3 Framework,
G.Eulisse, CHEP 2019: T6
https://doi.org/10.1051/epjconf/202024506032

- Apache Arrow as backing store for the message passing.
- Arrow fits well to represent column oriented data, while providing some level of flexibility for nested data via the usual record shredding.
- Using Apache Arrow allows for seamless integration with a larger ecosystem of tools, like Pandas or TensorFlow.

Online Reconstruction: O2 Facility

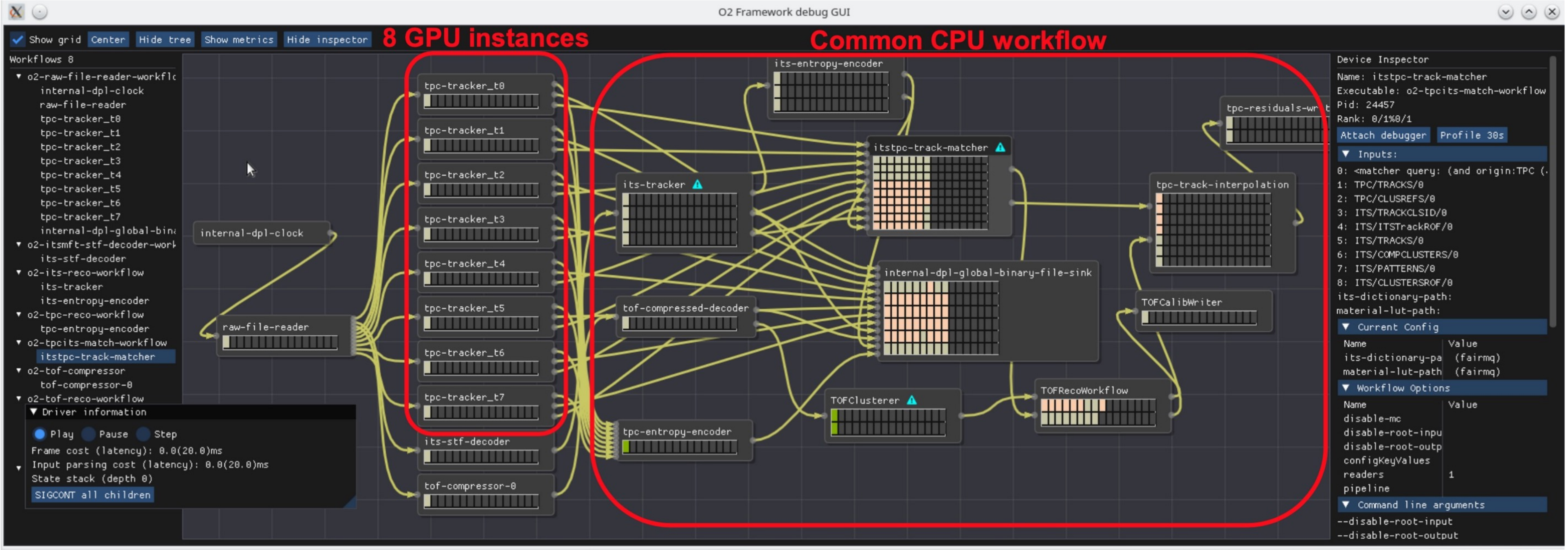


Baseline correction, zero suppression cluster finder.

Data volume reduction by online tracking, Asynchronous processing

Data Storage

ALICE EPN workflow



D. Rohr, G. Eulisse

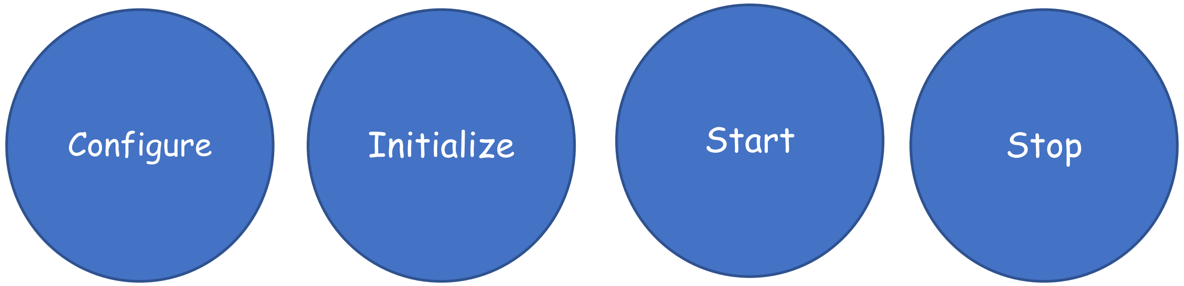
Controlling Processing graphs!

Controlling FairMQ state machine: on one device:

- The FairMQ core library provides two device controllers
  - Static : a fixed sequence of state transitions
  - Interactive: a read-eval-print-loop which reads keyboard commands from standard input
- A device controller only knows how steer a single FairMQ device (i.e: it runs in a thread within the device process)

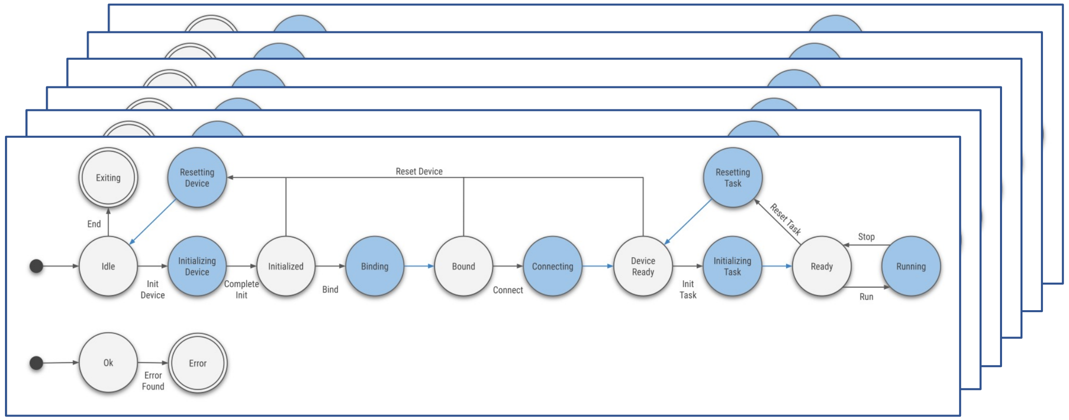
Controlling FairMQ state machine on a processing farm

- One has to make the entire cluster state available for the experiment control system and not single process one



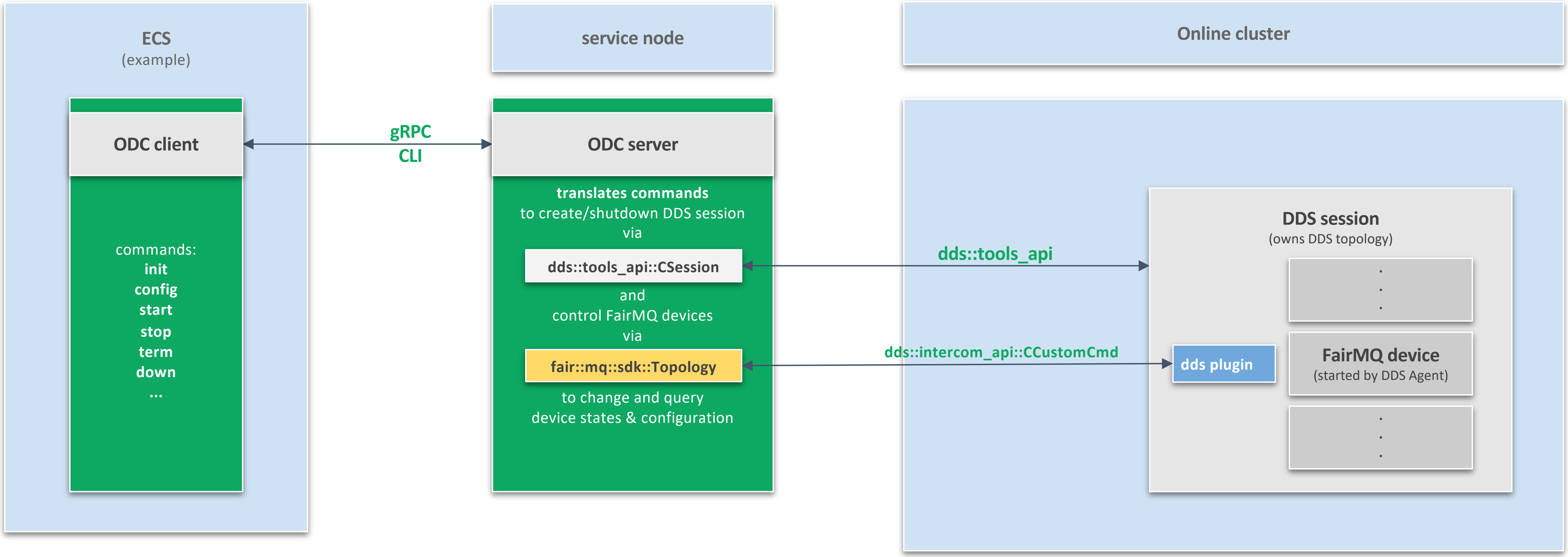
Exported cluster state machine

EPNs internal state machine (FairMQ)



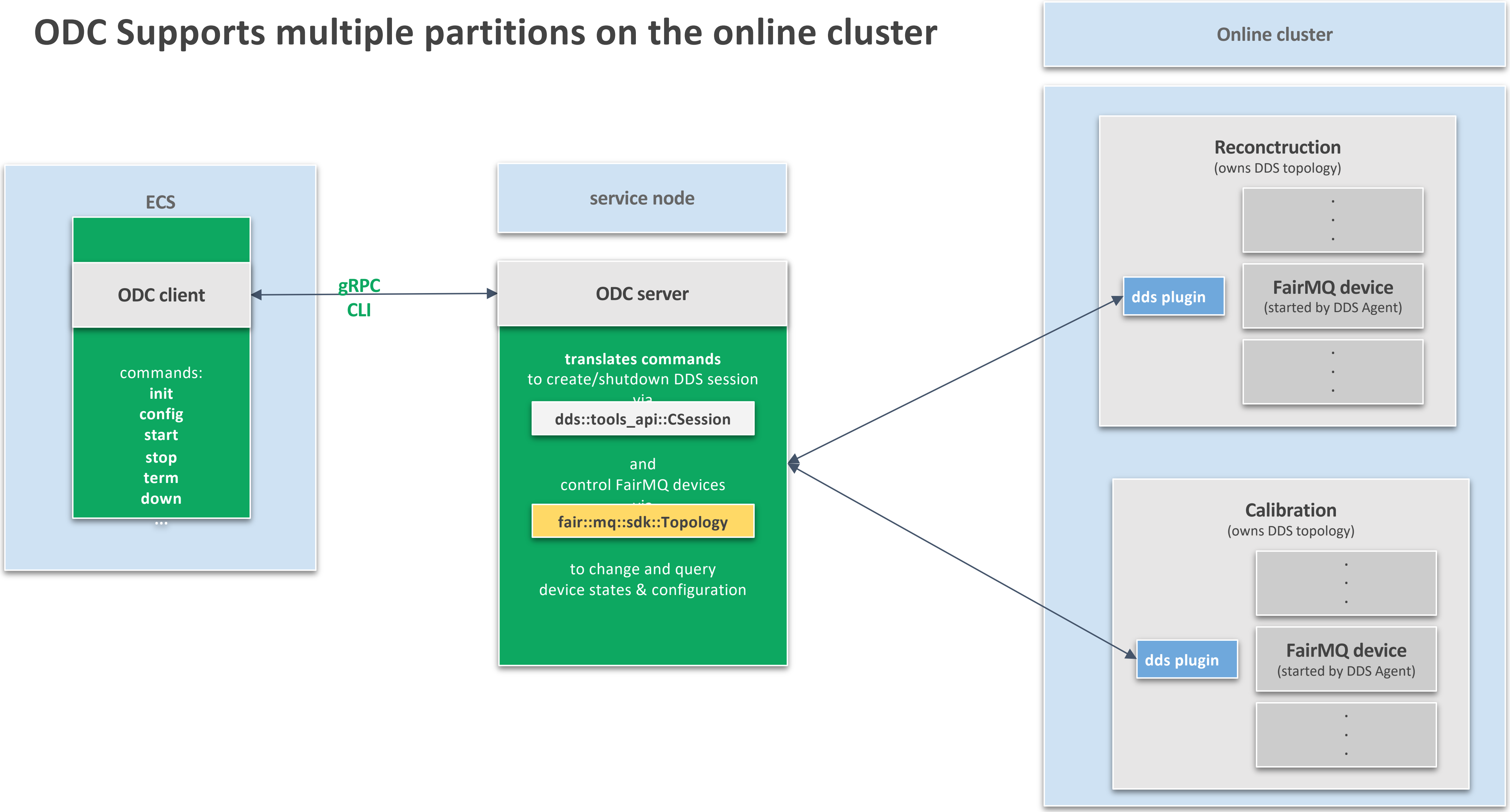
Controller (Architecture)

FairMQ SDK allows to **remotely configure** (set and get properties) **and control** (steer the state machine) a (sub-)set of **FairMQ devices** in a given topology. Communicates with the control/config plugin of the device. Set of asynchronous and synchronous APIs.



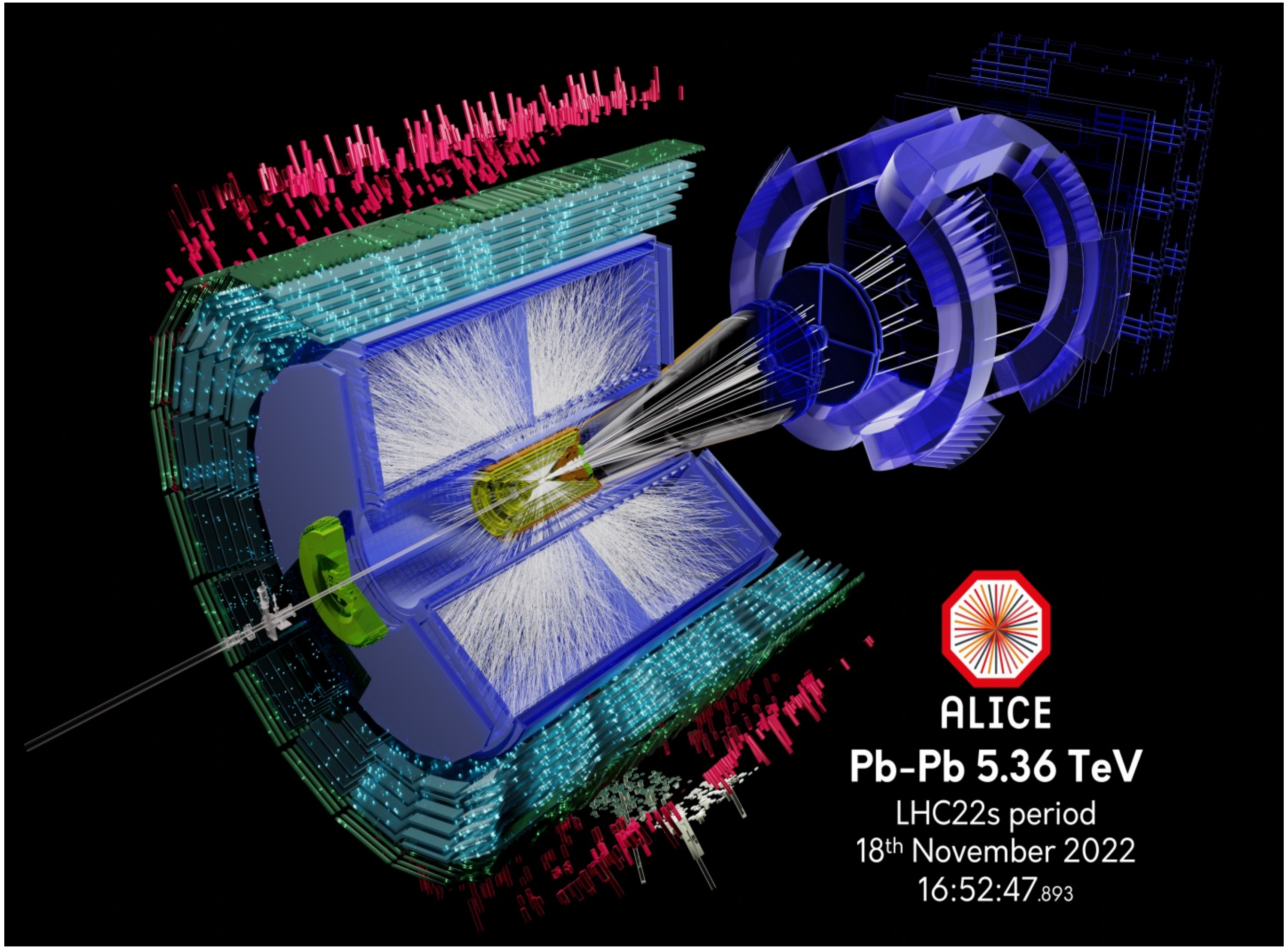
Online device controller (Architecture)

ODC Supports multiple partitions on the online cluster



Online device controller in RUN3

The online Device Controller (ODC) software developed at the GSI has been used to deploy and control ~70 000 tasks on 200 nodes (800 GPUs and over 20 000 CPUs) in the online farm directly connected to the ALICE detector.



Summary

- ALFA allows developers to write their specific code in whatever language they choose as long as that language can send and receive data through message queues.

- allows non-expert to write messaged based code without going into the details of the transport or the system below

Summary (continued)

- offers a **clean** and **maintainable** and **extendable** interface to the existing different data transport (ZMQ, nanomsg, shared Memory, OFI, ..etc)

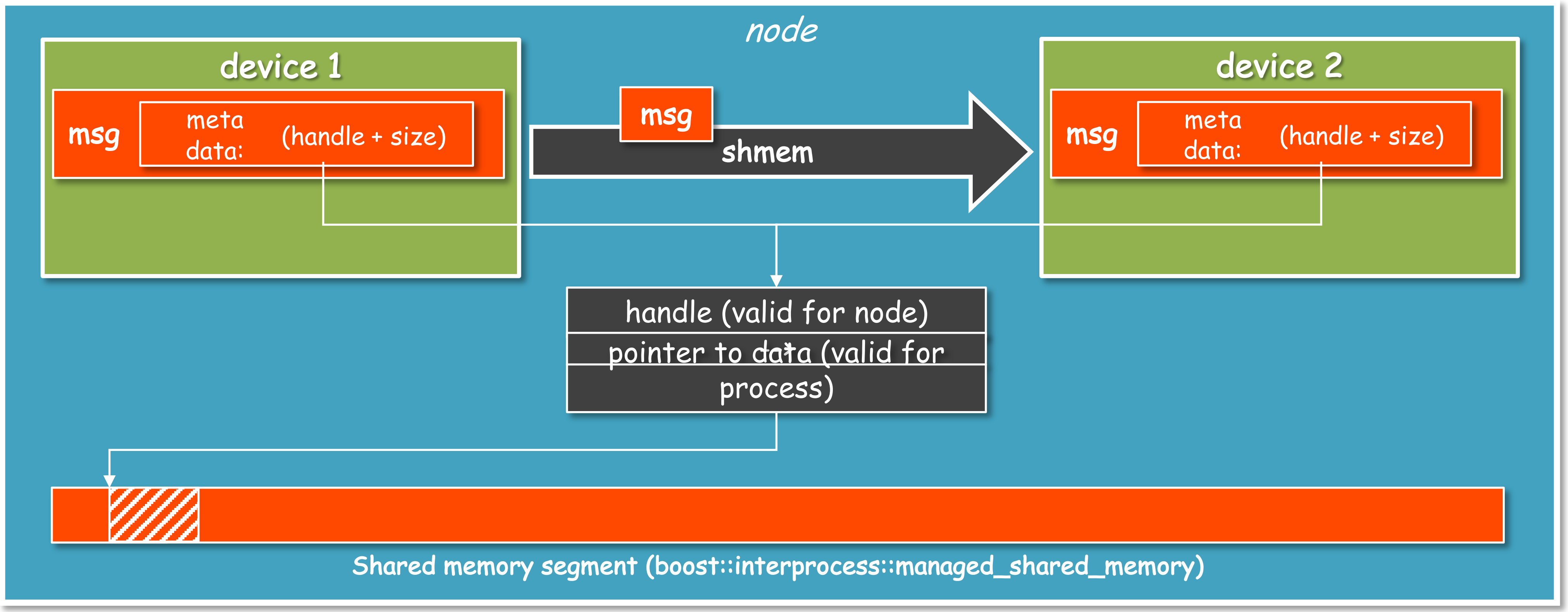
- provides utilities to deploy and control topologies on computing clusters, online clusters as well as on a laptop

Backup

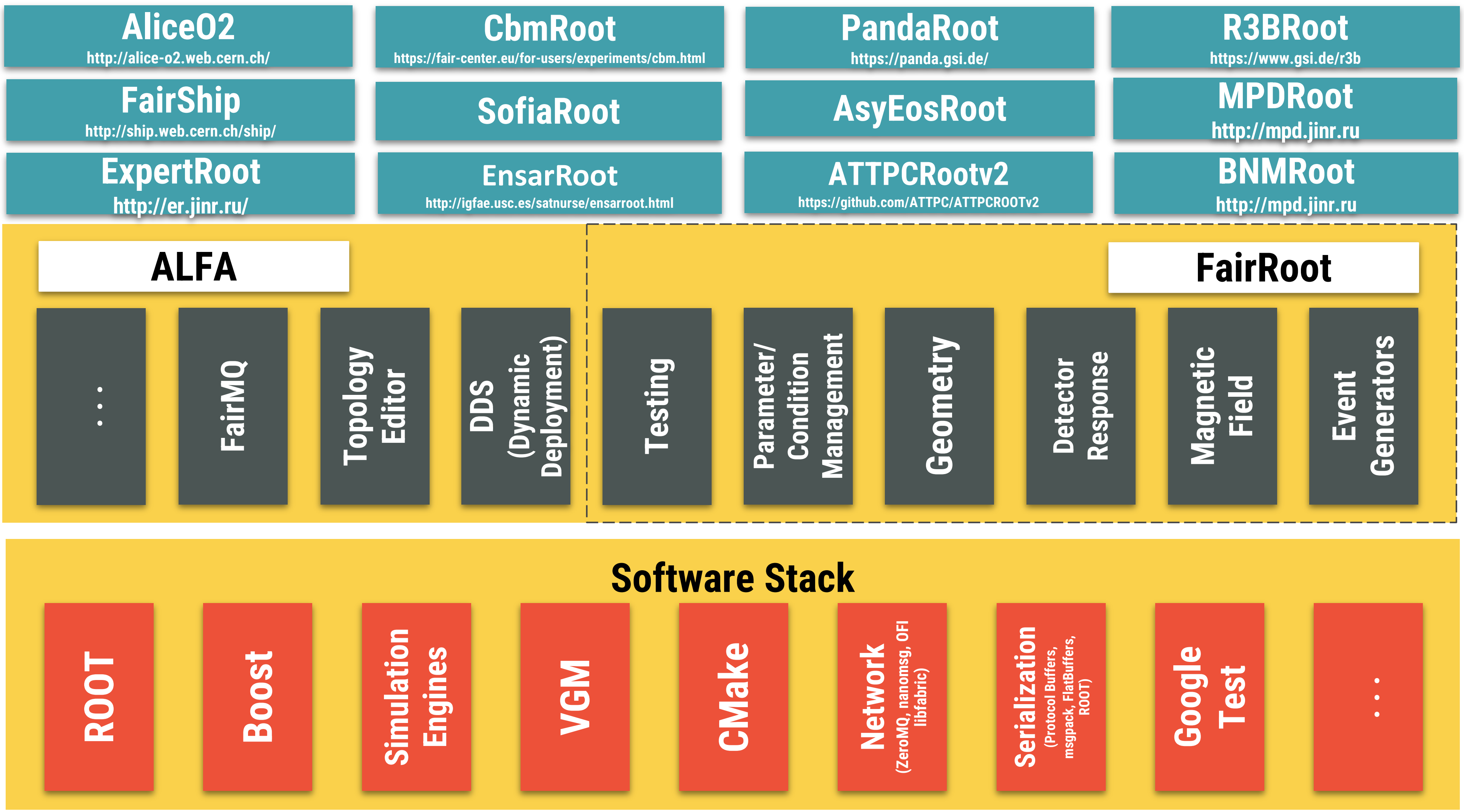
FairMQ Transport: Ownership

- Message owns data.
- Sender device (user code) passes ownership of data to framework with send call.
- Framework transfers to next device, passes ownership to receiver (no physical copy of the data with shared memory transport).
- No sharing of ownership between different devices – if the same message is needed by more than one receiver it is copied.

FairMQ Shared Memory Transport



FairRoot



AliceO2
http://alice-o2.web.cern.ch/
CbmRoot
https://fair-center.eu/for-users/experiments/cbm.html
PandaRoot
https://panda.gsi.de/
R3BRoot
https://www.gsi.de/r3b
FairShip
http://ship.web.cern.ch/ship/
SofiaRoot
AsyEosRoot
MPDRoot
http://mpd.jinr.ru
ExpertRoot
http://er.jinr.ru/
EnsarRoot
http://igfae.usc.es/satnurse/ensarroot.html
ATTPCRootv2
https://github.com/ATTPC/ATTPCROOTv2
BNMRoot
http://mpd.jinr.ru
ALFA
FairRoot
. . .
FairMQ
Topology Editor
DDS (Dynamic Deployment)
Testing
Parameter/ Condition Management
Geometry
Detector Response
Magnetic Field
Event Generators
Software Stack
ROOT
Boost
Simulation Engines
VGM
CMake
Network (ZeroMQ, nanomsg, OFI libfabric)
Serialization (Protocol Buffers, msgpack, FlatBuffers, ROOT)
Google Test
. . .

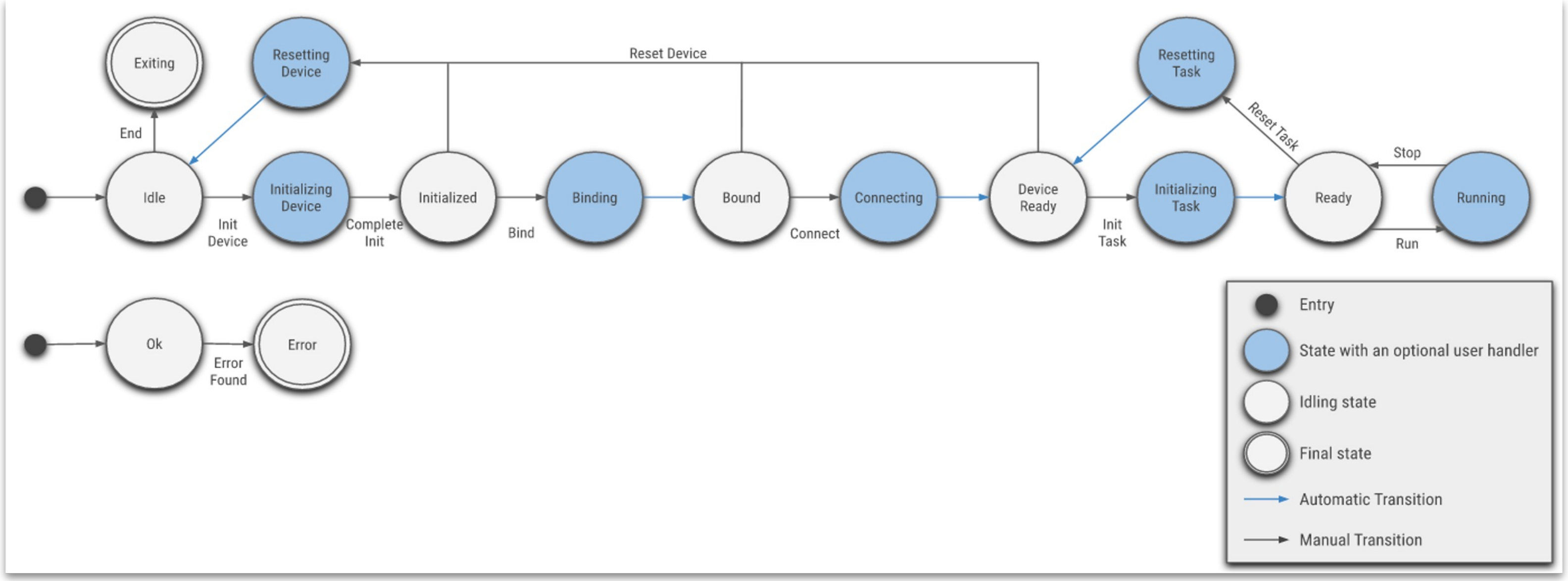
Software development for Experemints (SDE) Group

Projects:

- FairRoot: fairroot.gsi.de
- FairMQ: github.com/FairRootGroup/FairMQ
- DDS: dds.gsi.de
- ODC: github.com/FairRootGroup/ODC
- VC: github.com/VcDevel/Vc

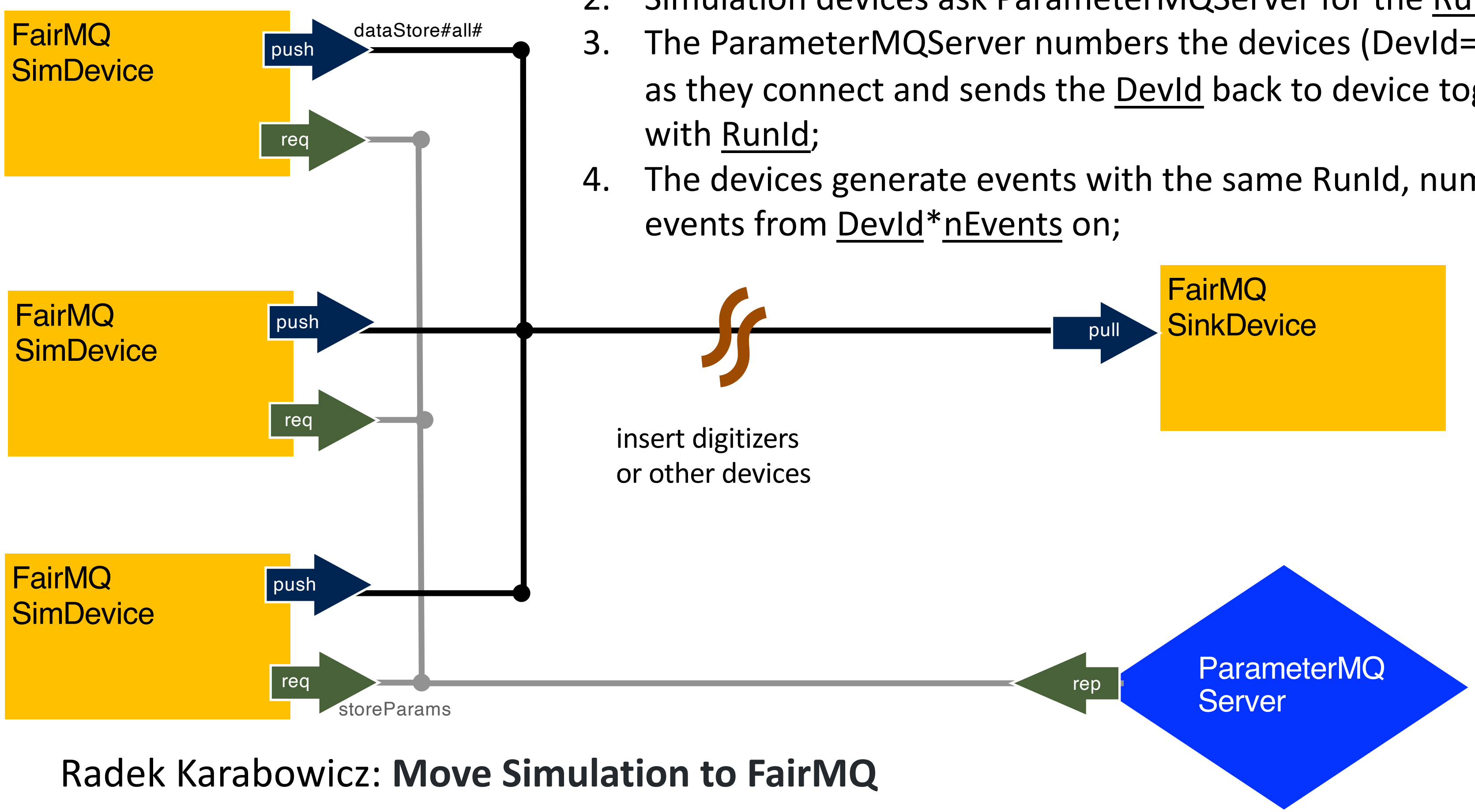
FairMQ State Machine & Example ECS Command Mapping

ECS command	DDS/FairMQ actions
init	DDS: Create session, submit agents, activate topology -> devices go in Idle state
configure	Devices: InitDevice->CompleteInit->Bind->Connect->InitTask
start	Devices: Run
stop	Devices: Stop
term	Devices: ResetTask->ResetDevice->End
down	DDS: Shutdown session



Distributed Simulation with FairMQ

1. Each simulation device will generate nEvents;
2. Simulation devices ask ParameterMQServer for the RunId;
3. The ParameterMQServer numbers the devices (DevId=0,1,2,…) as they connect and sends the DevId back to device together with RunId;
4. The devices generate events with the same RunId, numbering events from DevId*nEvents on;



Radek Karabowicz: **Move Simulation to FairMQ**
https://github.com/FairRootGroup/FairRoot/tree/dev/examples/MQ/pixelDetector