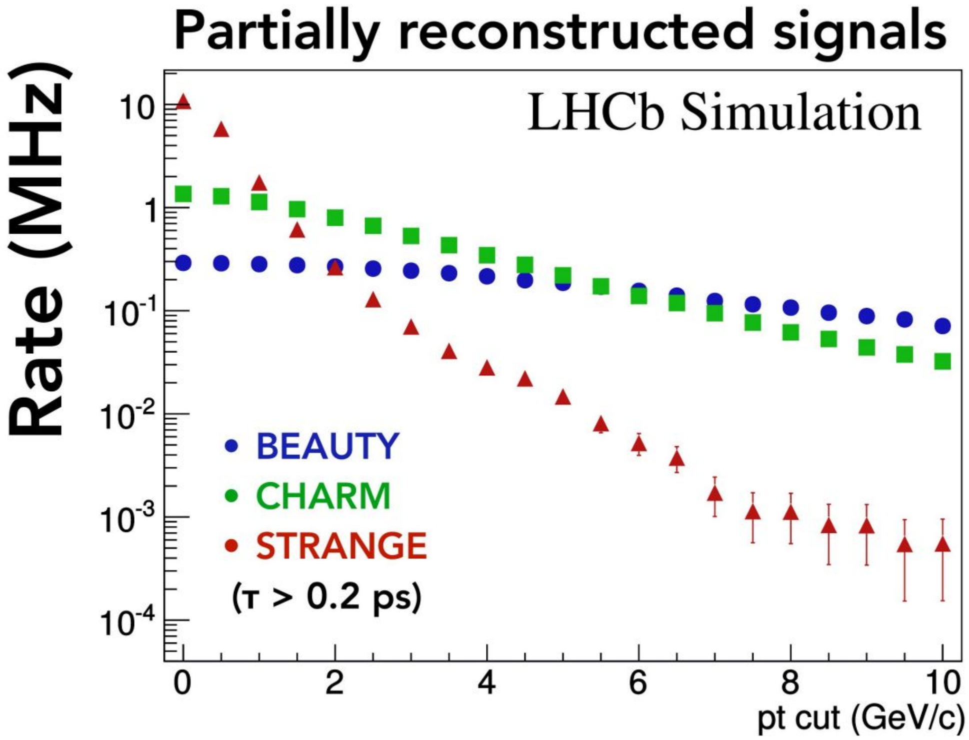
LHCb Upgrade Restart

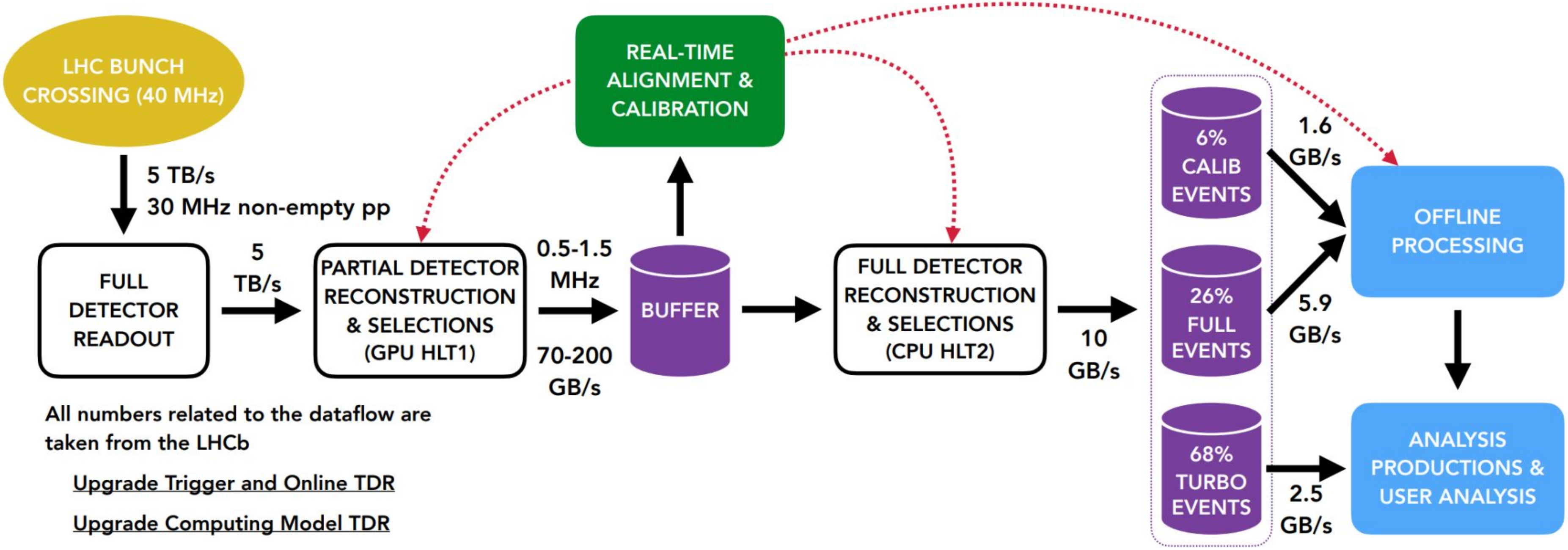
Software & Computing Round Table
5 April 2022
Rosen Matev

Why upgrade LHCb?



30 MHz (5 TB/s) of input contains a MHz of signal, while we can only store 10 GB/s long-term

LHCb upgrade dataflow

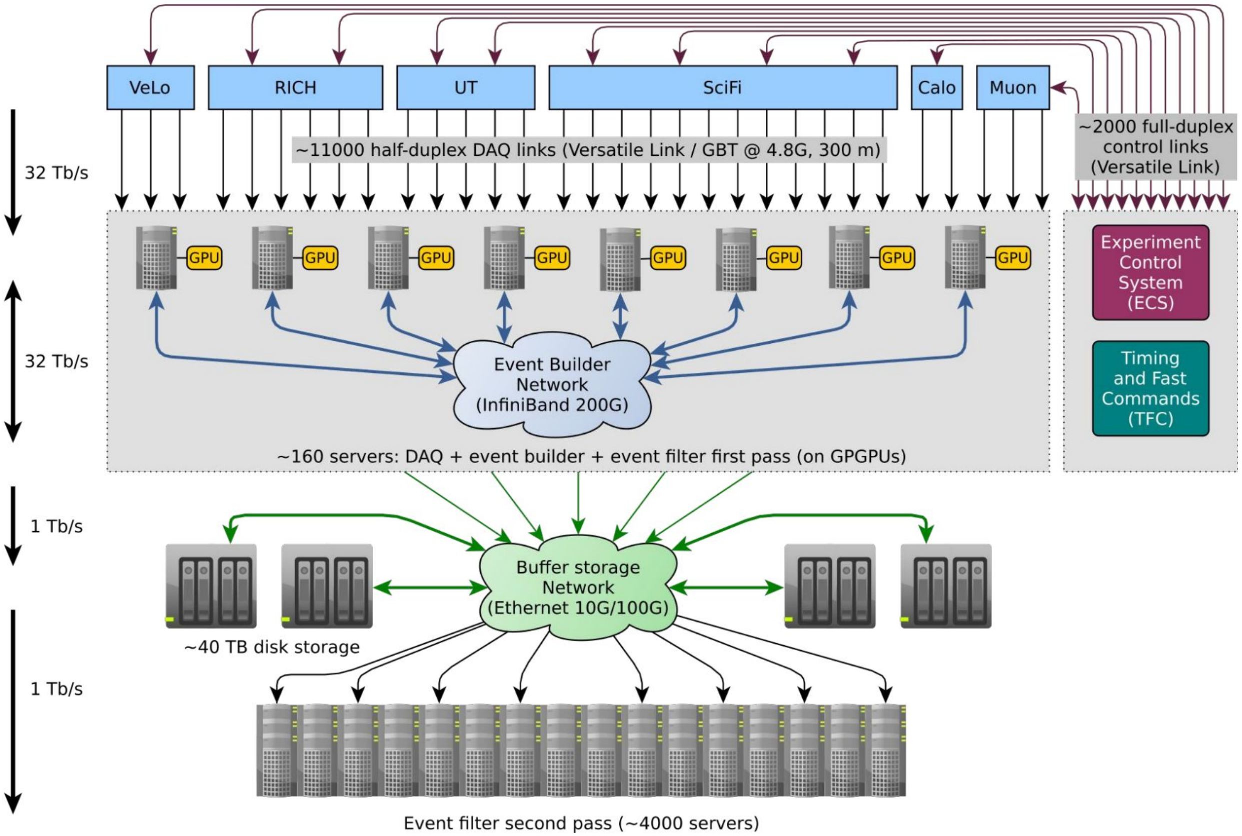


All numbers related to the dataflow are taken from the LHCb

Upgrade Trigger and Online TDR

Upgrade Computing Model TDR

DAQ architecture



30 MHz (5 TB/s) of event building and processing in a data center

LHCb software projects

- **Computing**: core software, offline productions, data management
- **Real Time Analysis (RTA)**: high-level trigger (reconstruction, selections, persistence), alignment and calibration
- **Data Processing & Analysis (DPA)**: slimming/trimming, tools for analysis
- **Simulation**: generators, transport, digitisation, ...
- (**Online**: primarily hardware, networking, storage, but also a lot of software)
- **Disclaimer: the following is heavily biased towards RTA**

Corollary: we don't only have "good code" when collisions begin

=> the better we understand the system the faster we can recover

=> tools, automation and clear processes are very important

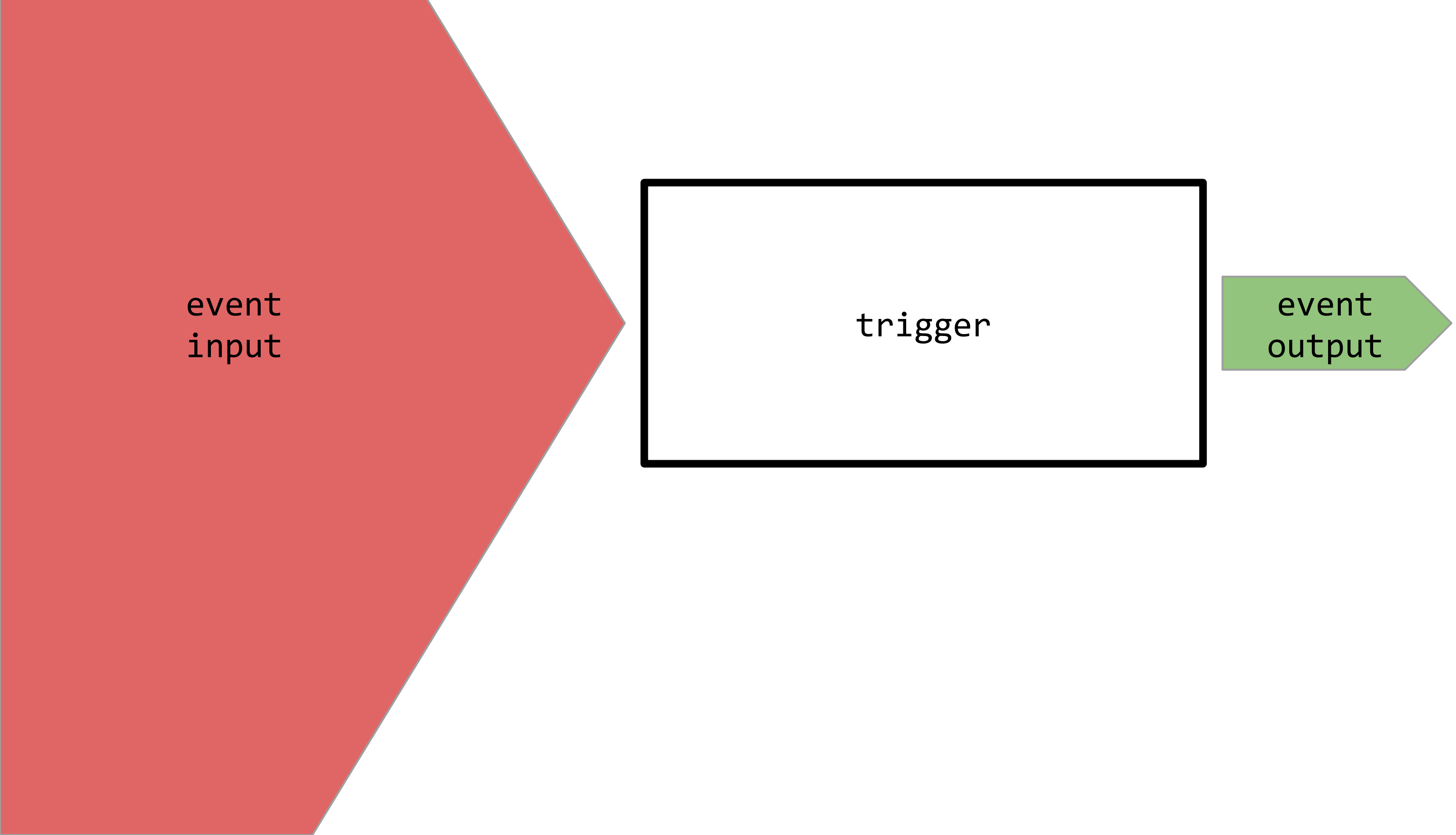
Release philosophy

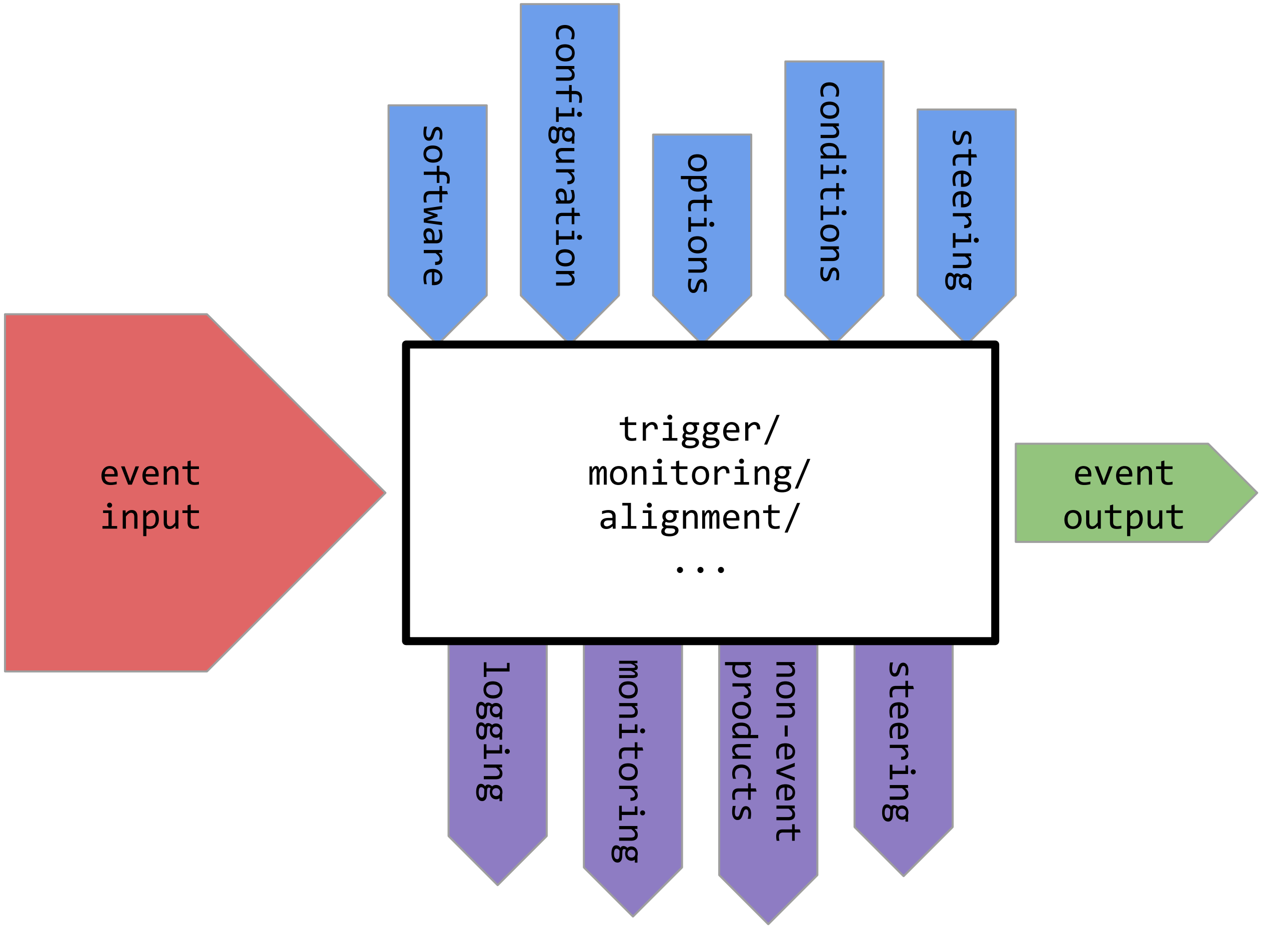
- In stable data taking (e.g. during Run 2)
  - cut a production branch each year (as late as possible)
  - only fixes and "localised" changes (e.g. new/tuned HLT2 exclusive selection)
- In a commissioning period (e.g. now):
  - maintain a single branch
  - most effort goes into the running system, lesser expectation of stable output
  - postpone working on and merging non-critical changes
- In any case, ready to release prod/main at any point
  - solid framework helps (e.g. "hard" to introduce thread-safety issues)
  - good test coverage (some unit tests, mostly integration tests)
  - code review that is appropriately thorough

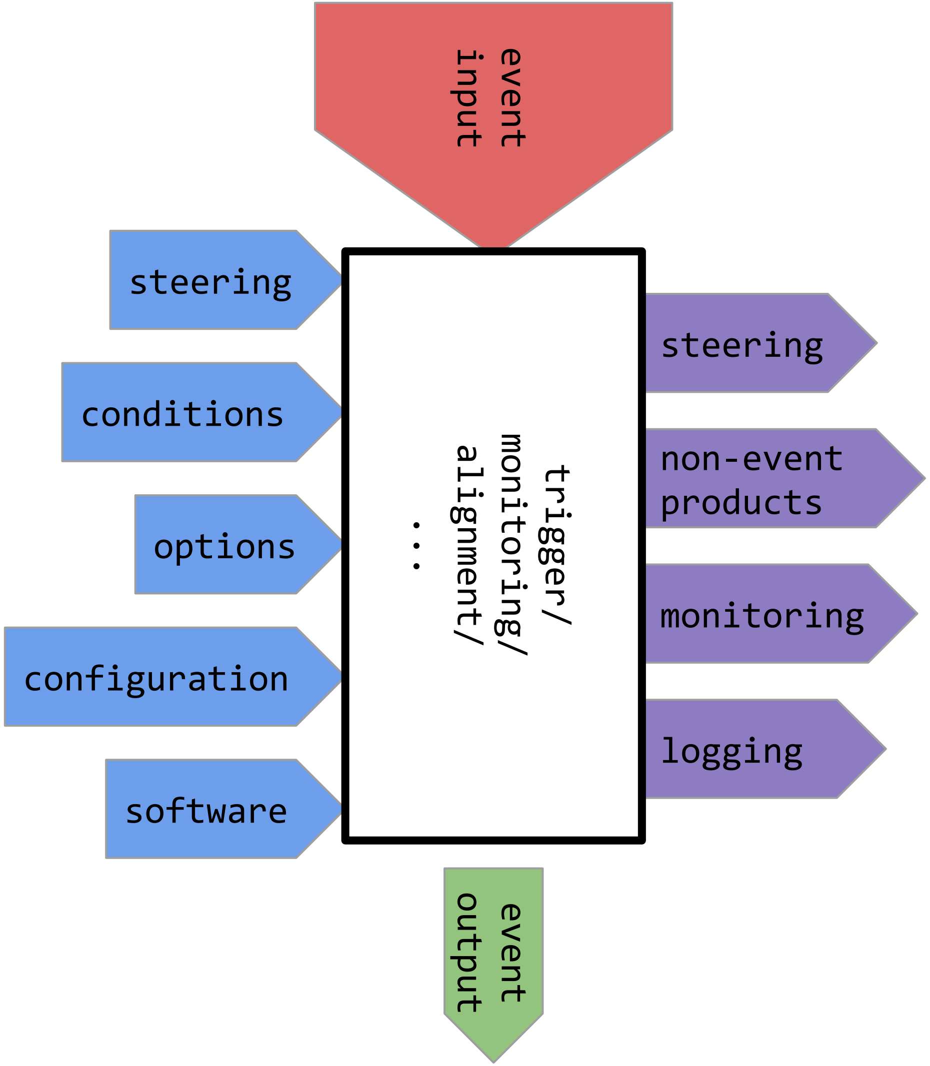
What do we test

- All of these usually happen on every merge request (GitLab)
- "Nightly" build system
  - compilation and unit tests
  - integration tests on O(1-1000) events
  - functional tests of e.g. persistence
  - any change in performance (within 1e-4) needs to be "blessed"
- Performance and regression "LHCbPR" tests
  - can take longer or may need dedicated resources
  - e.g. throughput tests, reconstruction performance, rate/efficiency tests
  - flag significant changes in computing or physics performance

Running Gaudi online




software
configuration
options
conditions
steering
event
input
trigger/
monitoring/
alignment/
...
event
output
logging
monitoring
non-event
products
steering


event input
steering
conditions
options
configuration
software
trigger/ monitoring/ alignment/ ...
steering
non-event products
monitoring
logging
event output

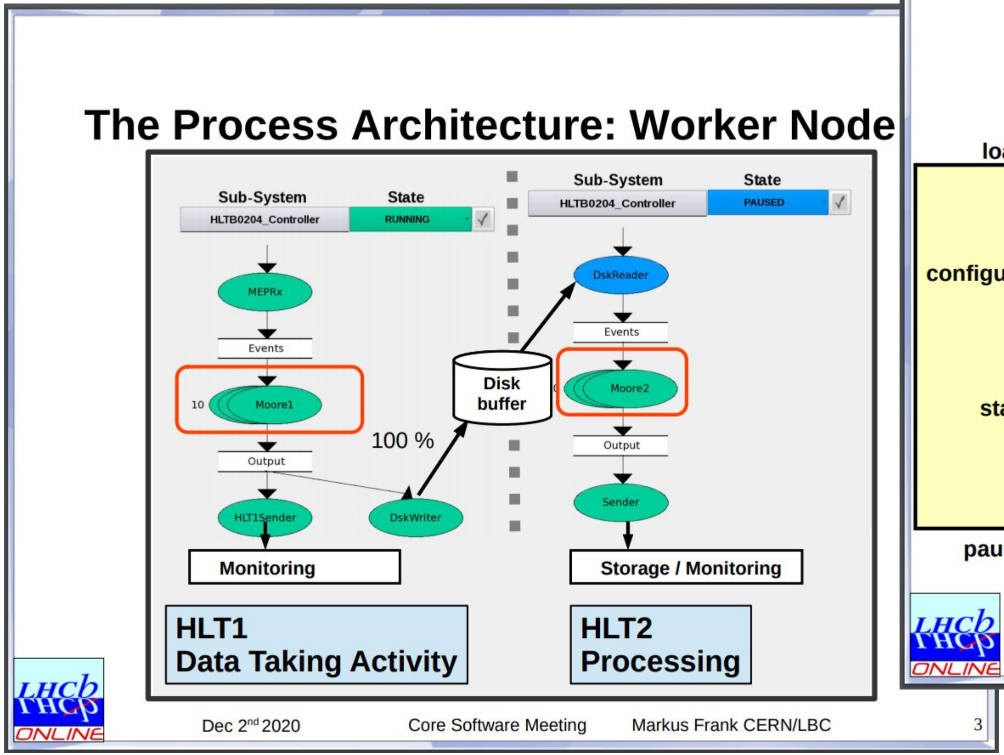
How do we run this thing?

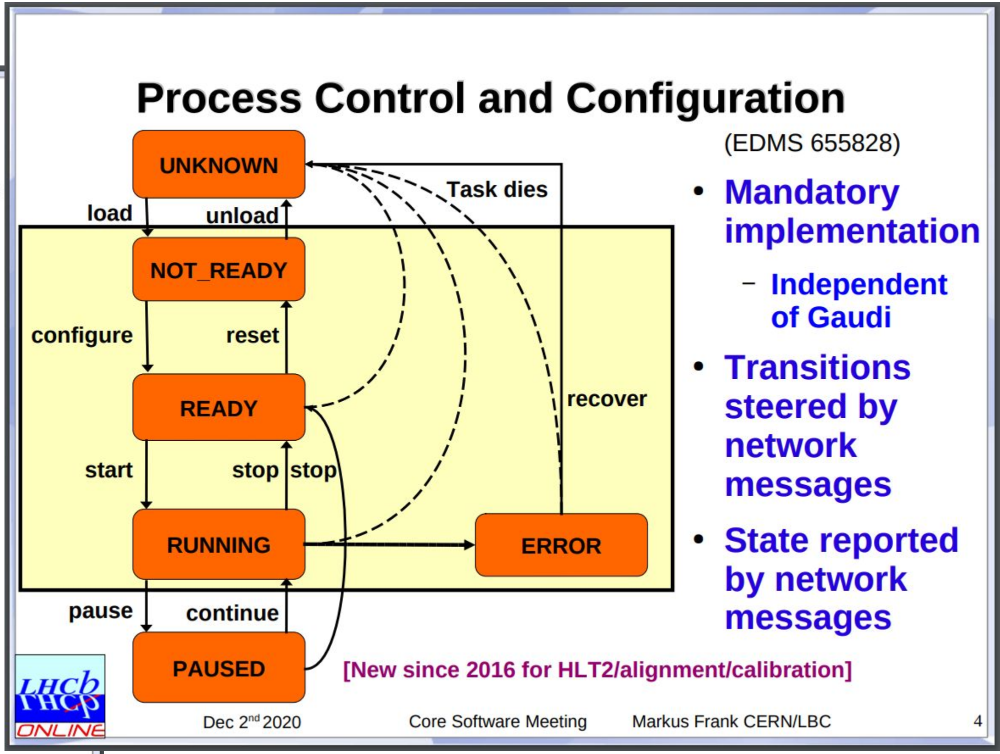
- Release and deploy software on CVMFS
- Create and deploy “trigger configuration” DBs
- Install online, update the WinCC-based run control
- Check that it configures
- 🤞 and wait for stable beams
- Look at some plots and logs
- Debug
- Rinse and repeat

Non-event-data inputs

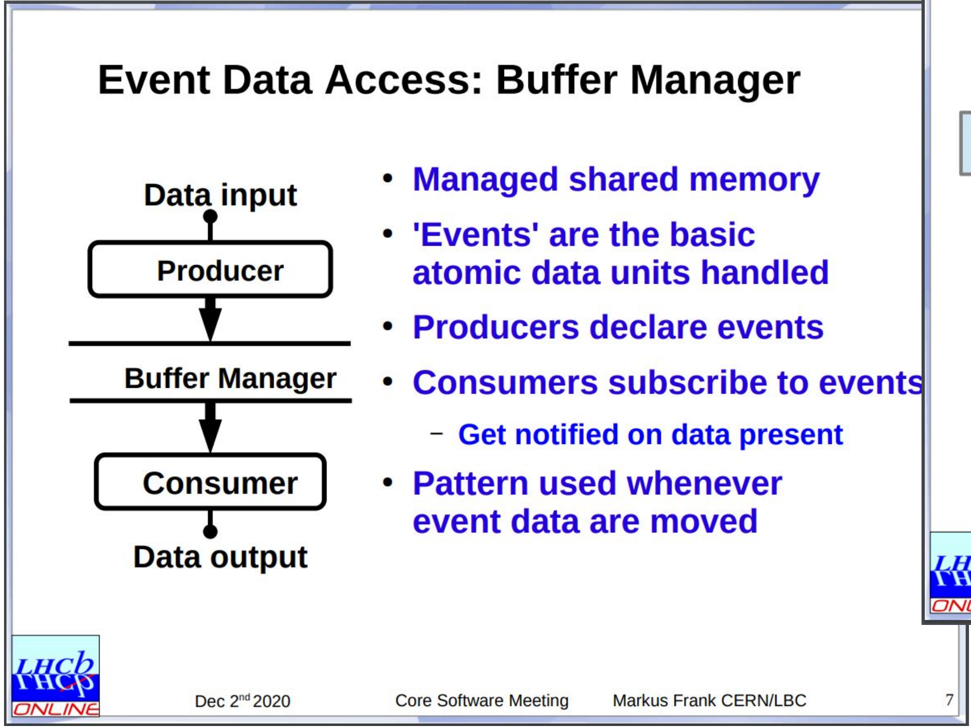
- Software: mostly CVMFS (local cache) + a bit of NFS
  - Run 2: ~50k processes on 1.5k nodes reading from NFS at the same time: slow
  - now: better NFS, fewer processes (multi-threading)
- Configuration DBs (trigger config, MVA weights, ...): CVMFS
- Run control options, conditions: NFS
  - run control generates files, tasks read them
  - pros: very simple, easy to debug
  - cons: potential contention point, potential sync issues
- Steering: network socket

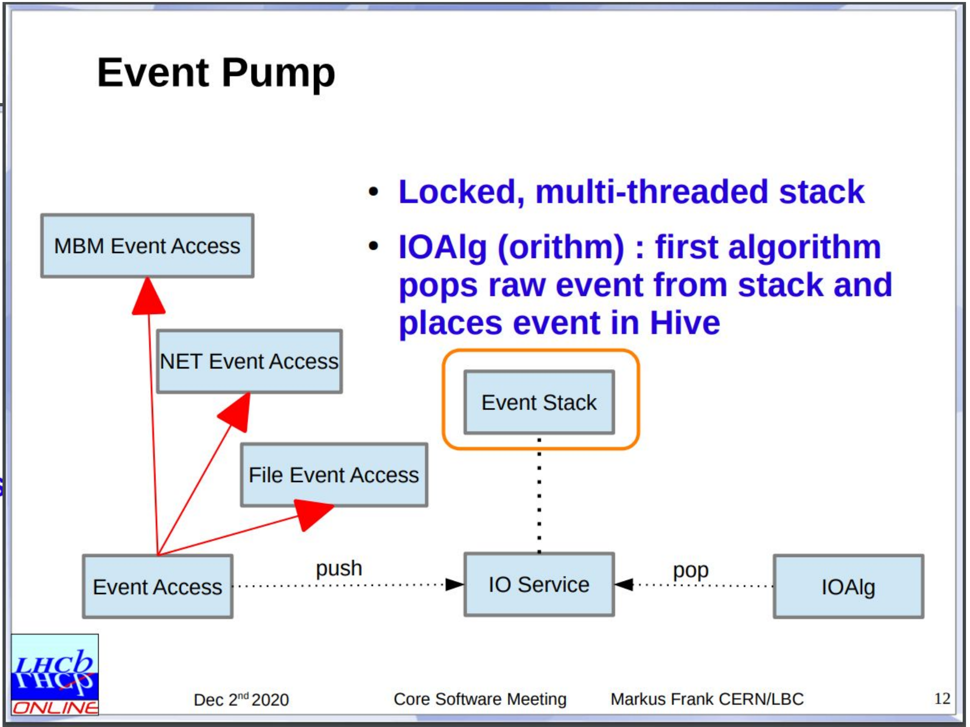
Online integration (steering)





Online integration (data)



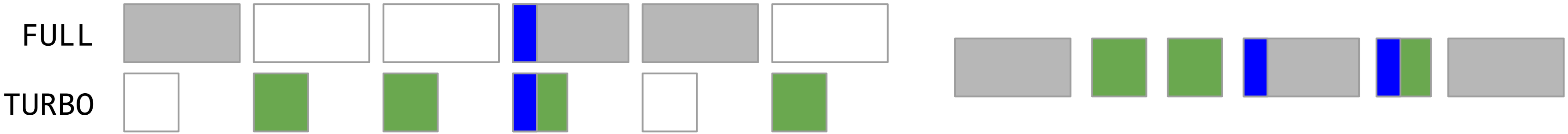


Non-event-data outputs (monitoring)

- Logging is a kind of monitoring
  - nominally used for debugging only; avoid having “expected” messages
- Several reasons why we need monitoring
  - quality control (is trigger config okay) + data quality (is data okay)
  - debugging issues (e.g. misconfiguration, performance)
  - real-time data for LHC, record of conditions (e.g. inputs for MC)
- Several ways to get stuff out
  - over the network (DIM) for histograms and counters
  - plain files (conditions)

Trigger output

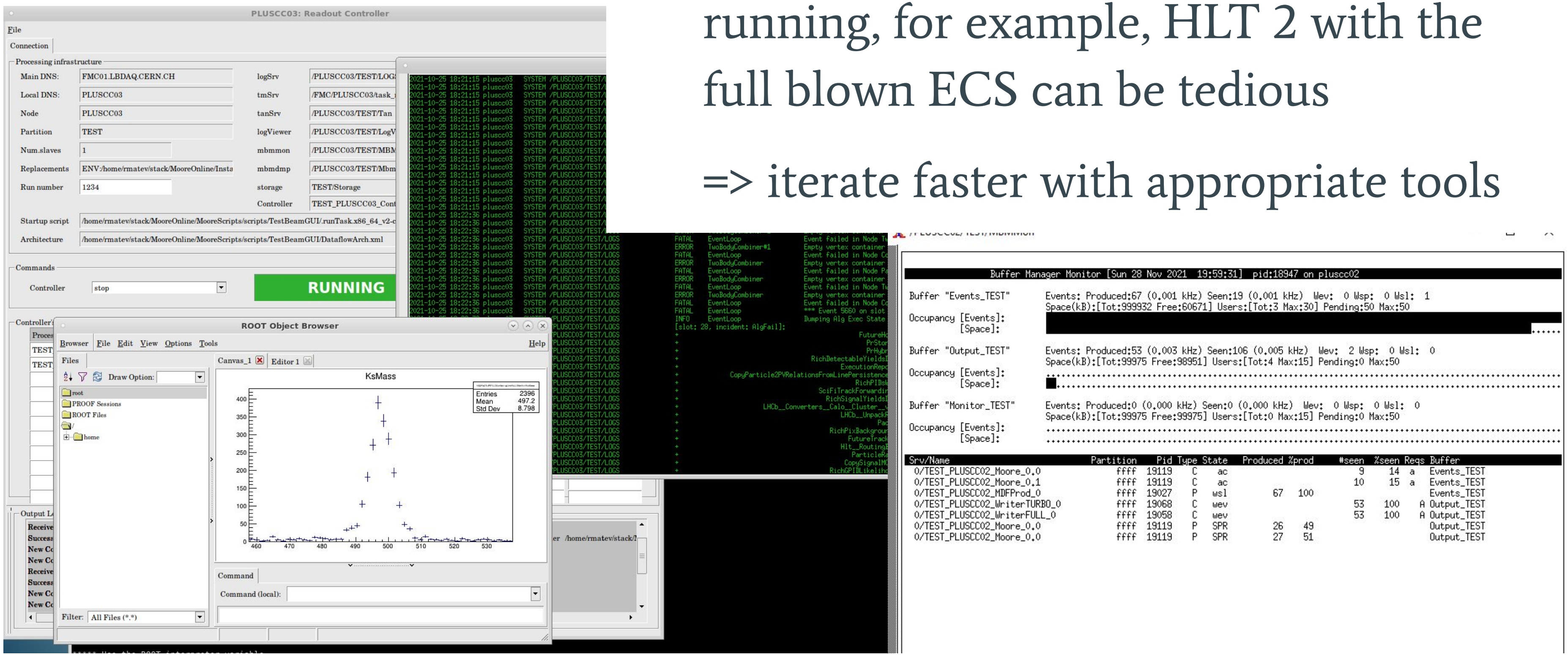
- Dedicated raw data format in the online system
  - trivial and concatenable (after HLT1)
- “Routing bits” are set by HLT1 / HLT2 per event
  - used to decide which event goes where (e.g. input of monitoring tasks or data for tracker alignment, etc.)
- HLT 2 needs to stream internally (different content per stream)



Online testbench

running, for example, HLT 2 with the full blown ECS can be tedious

=> iterate faster with appropriate tools

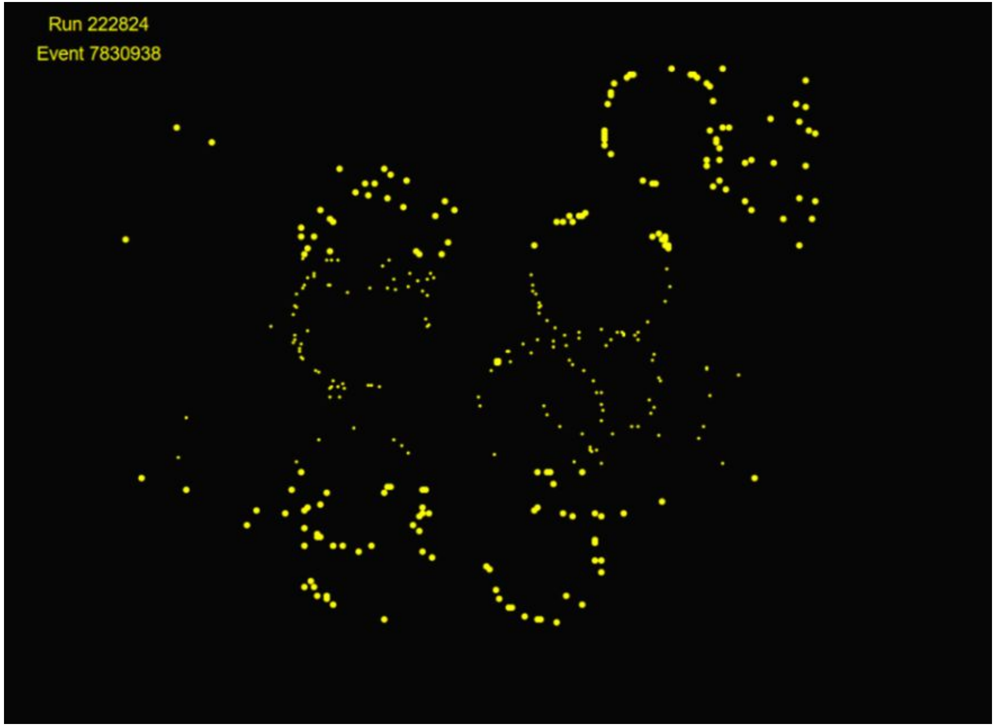


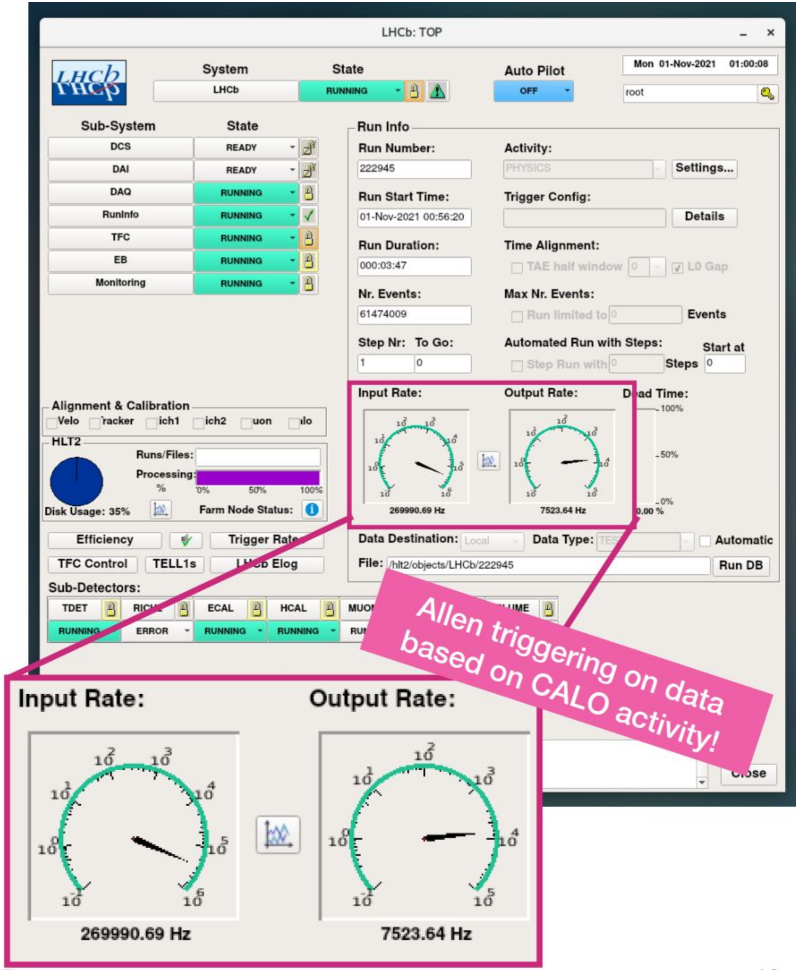
HLT1 (GPU trigger) integration

- Process data in the "SOA" layout provided by the event builder
  - Batches of 30k grouped by readout unit (multiple frontends), not by event
  - Process in batches of 1k events
- Wrapped into Gaudi (sans the event loop)
  - Steering by the experiment control system
  - Obtain geometry and conditions from "regular" stack on the fly
  - Deal with changing conditions
  - Monitoring output goes via the common service
- HLT1 hardware and processes share the server with event building: keep a close eye on CPU and memory usage

October`21 LHC test beam

- LHC performed excellently: they often go faster than expected!
- LHCb ran with the upgraded RICH, calorimeters and muon stations, and the new PLUME detector for the first time.
- Overall, an overwhelming success for the LHCb commissioning
  - online system commissioned and was demonstrated to work
  - participating detectors time aligned within a few days
  - the monitoring system / viz tool was integrated and used
  - HLT1 was progressively deployed on CPU/GPU in passthrough/activity mode
  - The full system was tested in «operation mode» with a shifter taking charge


Run 222824
Event 7830938


LHCb: TOP
Mon 01-Nov-2021 01:00:08
System
State
Auto Pilot
LHCb
RUNNING
OFF
Sub-System
State
DCS
READY
DAI
READY
DAQ
RUNNING
RunInfo
RUNNING
TFC
RUNNING
EB
RUNNING
Monitoring
RUNNING
Run Info
Run Number:
222945
Activity:
PHYSICS
Settings...
Run Start Time:
01-Nov-2021 00:56:20
Trigger Config:
Details
Run Duration:
000:03:47
Time Alignment:
Nr. Events:
61474009
Max Nr. Events:
Step Nr:
To Go:
Automated Run with Steps:
Input Rate:
Output Rate:
269990.69 Hz
7523.64 Hz
Disk Usage: 35%
Farm Node Status:
Efficiency
Trigger Rate
TFC Control
TELL1s
Data Destination:
Data Type:
Automatic
File:
/hlt2/objects/LHCb/222945
Run DB
Sub-Detectors:
TDET
ECAL
HCAL
RUNNING
ERROR
Allen triggering on data based on CALO activity!
Input Rate:
Output Rate:
269990.69 Hz
7523.64 Hz

Some takeaways

- Communication is crucial (with detectors and among software projects)
- Sharing infrastructure is really important:
  - detector monitoring tasks piggybacked on HLT2 integration work
- With software, you can do most of what you need remotely, however, you need such live sessions, because
  - **beam time is a strong motivator**: one year before beam, there's always something more important to do than figuring out the online integration
  - **it brings people physically together** (when there is no beam, organize "hackathons" or "commissioning weeks" to boost focus and get things done)
- Some example issues you'll only see with real data
  - data padding not implemented according to spec, irrelevant in MC
  - the channel map in the conditions does not correspond to the real cabling

More always needed!

- Testing
  - In Run 2 we only tested HLT 1 out-of-fill by running on random triggers.
  - HLT 2 could only be tested by running manually or risking data loss.
  - Deploying updates in Run 3 has to be fast!
  - Can we test in a reasonably large part of the system? In parallel to data taking?
- Easy to configure and run, profile and debug
  - The same configuration should run online and offline with a trivial switch
  - Send pathological events to a DEBUG stream
  - Automatic perf stats, save core dumps, …

Thank you

RTA's goals

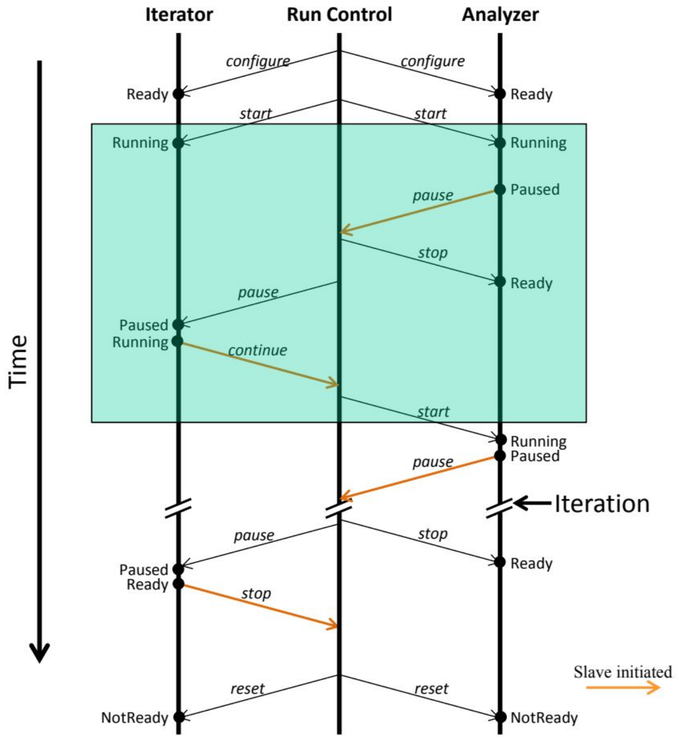
RTA's product is **physics data of high quality** obtained by means of

- software such as the trigger and alignment applications
- monitoring capability
- clear interpretation of data

As such we need to

- develop the trigger selections, reconstruction and calibration
- run the applications in the online / offline (MC) environment
- store data in a usable form and provide performance corrections

Online alignment FSM



- Each online alignment and calibration task is controlled by the same finite state machine
- One process of the analyser task runs on each of the ~1600 nodes in the trigger farm (in Run 2)
- Overview of sequence:
  - Iterator writes conditions in XML
  - Each analyser reads these conditions and reconstructs events to produce a binary file “alignsummarydata” (ASD)
  - Iterator combines the ASDs to compute the new conditions constants and writes these to XML
  - Steps 2 & 3 repeat until the procedure converges. The new constants are then copied to the trigger area.

Monitoring data flow