The Continuously Running iFDAQ of the COMPASS Experiment

M. Bodlak, V. Frolov, S. Huber, V. Jary, I. Konorov, A. Kveton, D. Levit, J. Novy, D. Steffen, **O. Subrt**, J. Tomsa, M. Virius

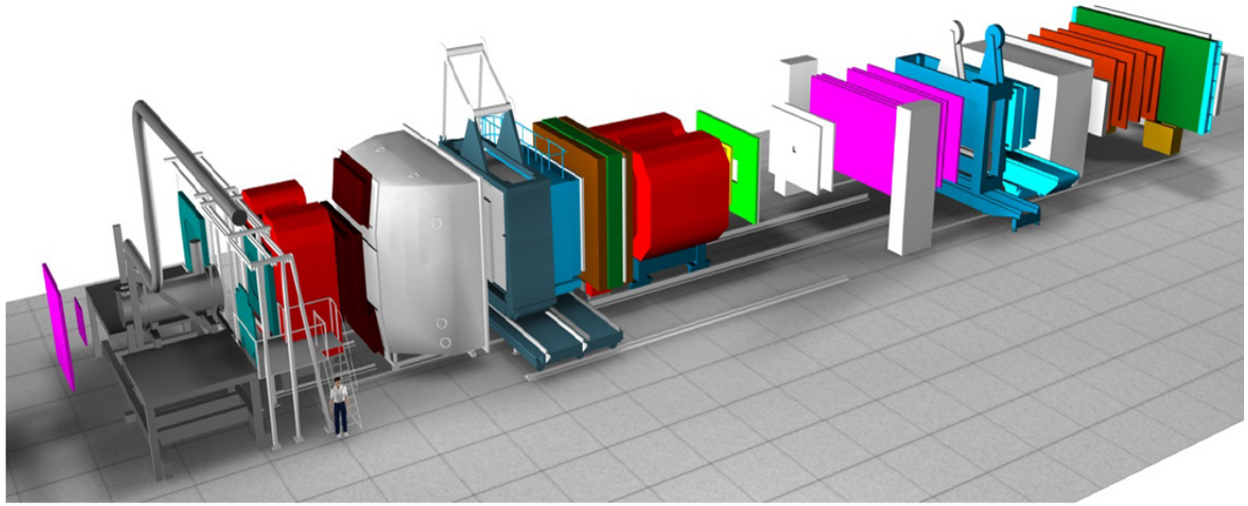
Faculty of Nuclear Sciences and Physical Engineering
Czech Technical University in Prague, Czech Republic
&
European Organization for Nuclear Research – CERN, Switzerland

Outline

- COMPASS experiment at CERN
- iFDAQ architecture
- DIALOG library
- DAQ Debugger
- iFDAQ stability
- Continuously running iFDAQ
  - Continuously Running Mode
  - Proper Timing and Synchronization

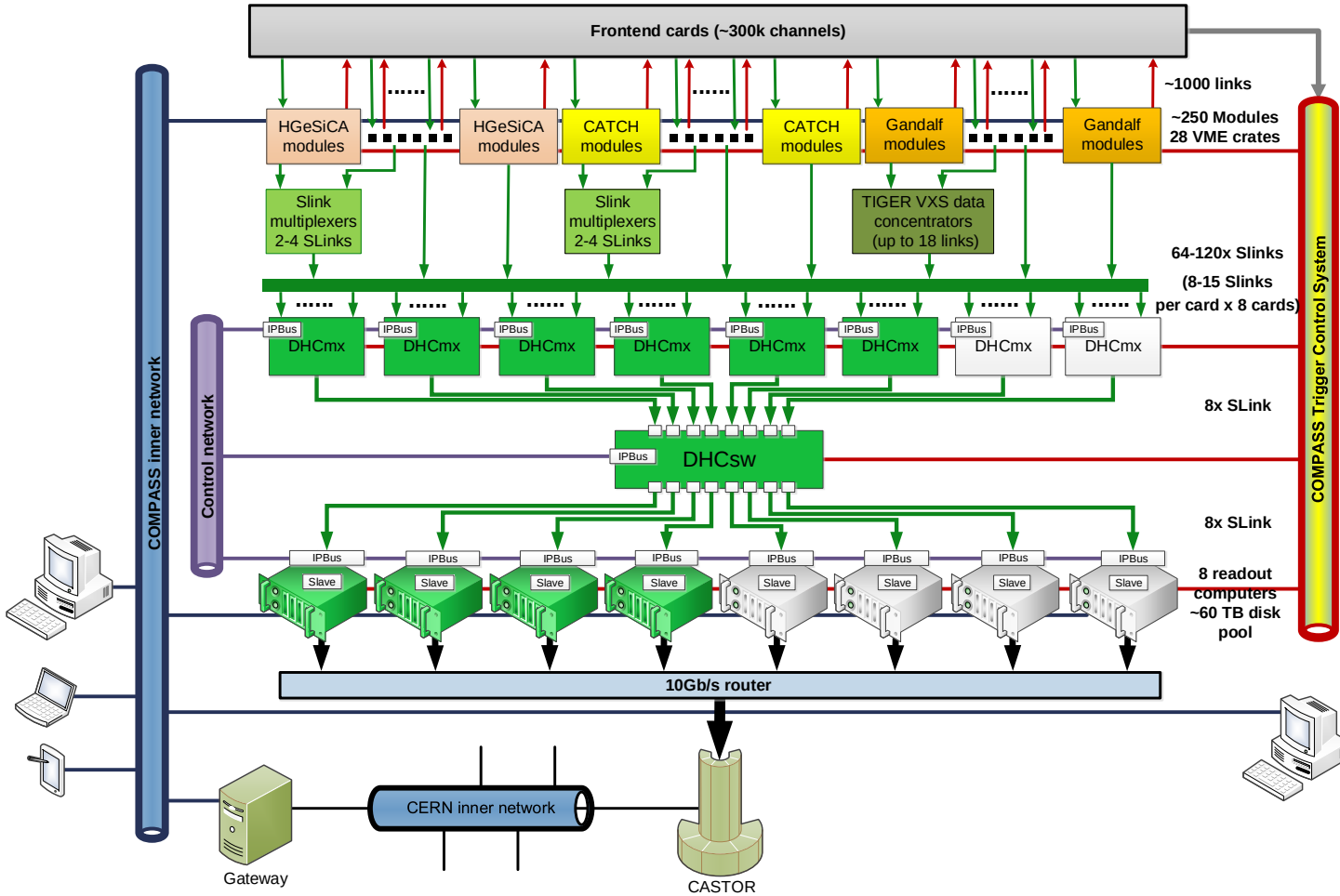
COMPASS Experiment

- Fixed target experiment at SPS accelerator at CERN
- Study of hadron structure and hadron spectroscopy with high intensity muon and hadron beams
- Data-taking started in 2002
- Trigger rate up to 40 kHz, average event size up to 50 kB
- In spill data rate 1.5 GB/s and sustained data rate 500 MB/s



Hardware Structure

- Hardware based E.B.
- Data concentrated by 6 (up to 8) DAQ modules with multiplexer firmware
- Distribution to 4 (up to 8) readout computers by DAQ module switch firmware
- Full events received by servers
- Consistency check at many layers
- Events checked and transferred to DATE data format

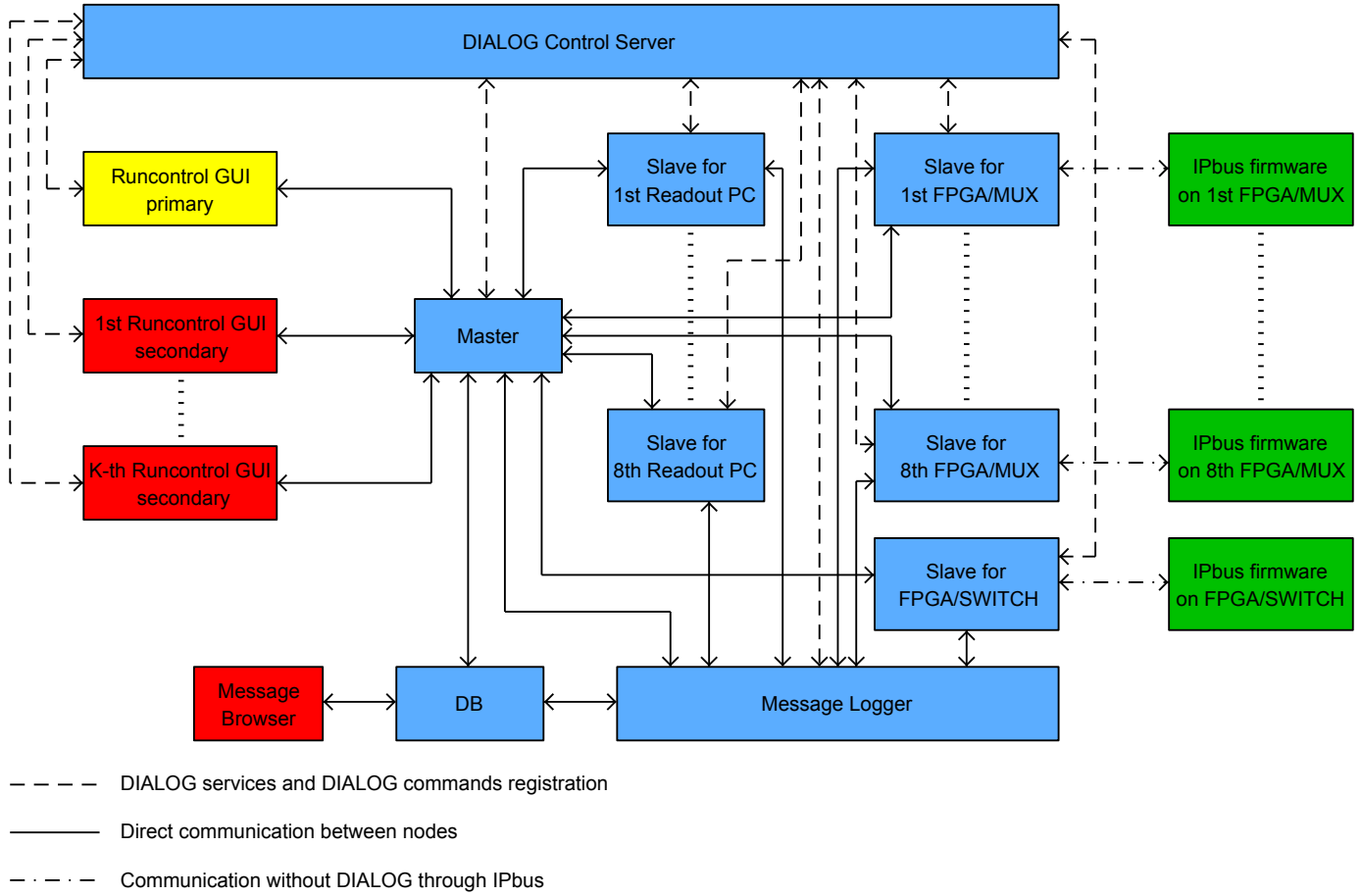


Used Software Technologies

- C++, Python
- Qt framework
- DIALOG library
- DAQ Debugger
- IPbus suite for communication with FPGA cards
- MySQL
- PHP, JavaScript
- Zabbix

Software Structure

- Runcontrol GUI is a graphical user interface
- Master is a main control process
- Slave-readout readouts and verifies the data
- Slave-control monitors and controls the FPGA cards
- MessageLogger stores informative and error messages into the database
- MessageBrowser provides an intuitive access to messages stored in the database



DIALOG Library

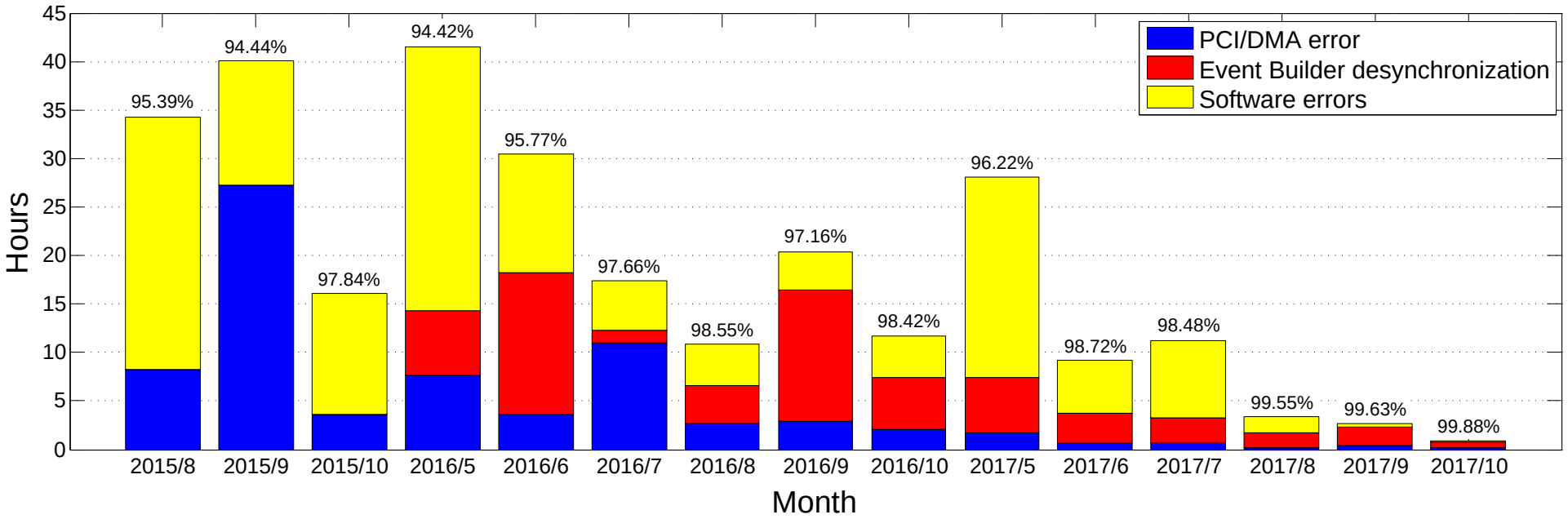
- New communication library for inter-process communication
- It is implemented in Qt framework
- Dialog means conversation, talk or speech (**D** – distributed, **I** – inter-process, **A** – asynchronous, **L** – library, **O** – open, **G** – general)
- Communication based on the publish/subscribe method
- It uses TCP/IP protocol and sockets for message transmission
- It saturates the 10 Gbps network bandwidth already with messages of size 1 kB

DAQ Debugger

- Library for the iFDAQ error detection
- Fully incorporated to all processes during the run 2016 and 2017
- Design requirements
  - The integration to running system requires interface for an easy use
  - It does not affect the process performance
  - It does not increase load on readout engine computers
  - It provides with reports in `/tmp` folder containing stack trace of all threads and memory dump
- Reports are investigated by iFDAQ experts
- Problem understanding $\rightarrow$ the fix is released and tested

iFDAQ Stability

- iFDAQ Software is stable since beginning of October 2017
- DIALOG helped to increase stability of the iFDAQ
- DAQ Debugger detected all remaining software issues
- iFDAQ UpTime – time when iFDAQ is able to take data
- iFDAQ UpTime is around 99.88% $\simeq$ 1 hour loss / month of data-taking



Motivation – Before Continuously Running iFDAQ

- Unstable iFDAQ $\rightarrow$ shorter runs with less than 200 spills
- The waste of beam time consisted of two parts
  - Human factor (few spills) – shift crew had to start a new run
  - Technical limitations (3 spills) – synchronization of TCS and SPS super cycle
- In the best case, we lost $\sim$ 5% of spills between two runs
- Other problems (beam spikes, detector and FEE problems) $\rightarrow$ many short runs $\rightarrow$ it easily reaches $\sim$ 30% of lost spills
- iFDAQ stability improvement $\rightarrow$ FEE upgrades, DIALOG library and DAQ Debugger integration

Goal – Promising Continuously Running iFDAQ

- Stable iFDAQ $\rightarrow$ smooth data-taking nonstop, no lost data
- 24/7 regardless of nights, weekends or bank holidays for most of the calendar year
- No user intervention is needed $\rightarrow$ effect of inattentive shift crew is decreased
- It creates automatically runs with 200 spills
- It helps to collect more physics data

Continuously Running Mode – Overview

- Automatic transition between two consecutive runs must be smooth
- Data files from the previous run must be safely closed and new files for a new run has to be opened
- The run number has to be increased in a correct time
- Old records has to be filled and new ones has to be created in electronic logbook
- One main requirement – a proper synchronization

Continuously Running Mode – Dry Run

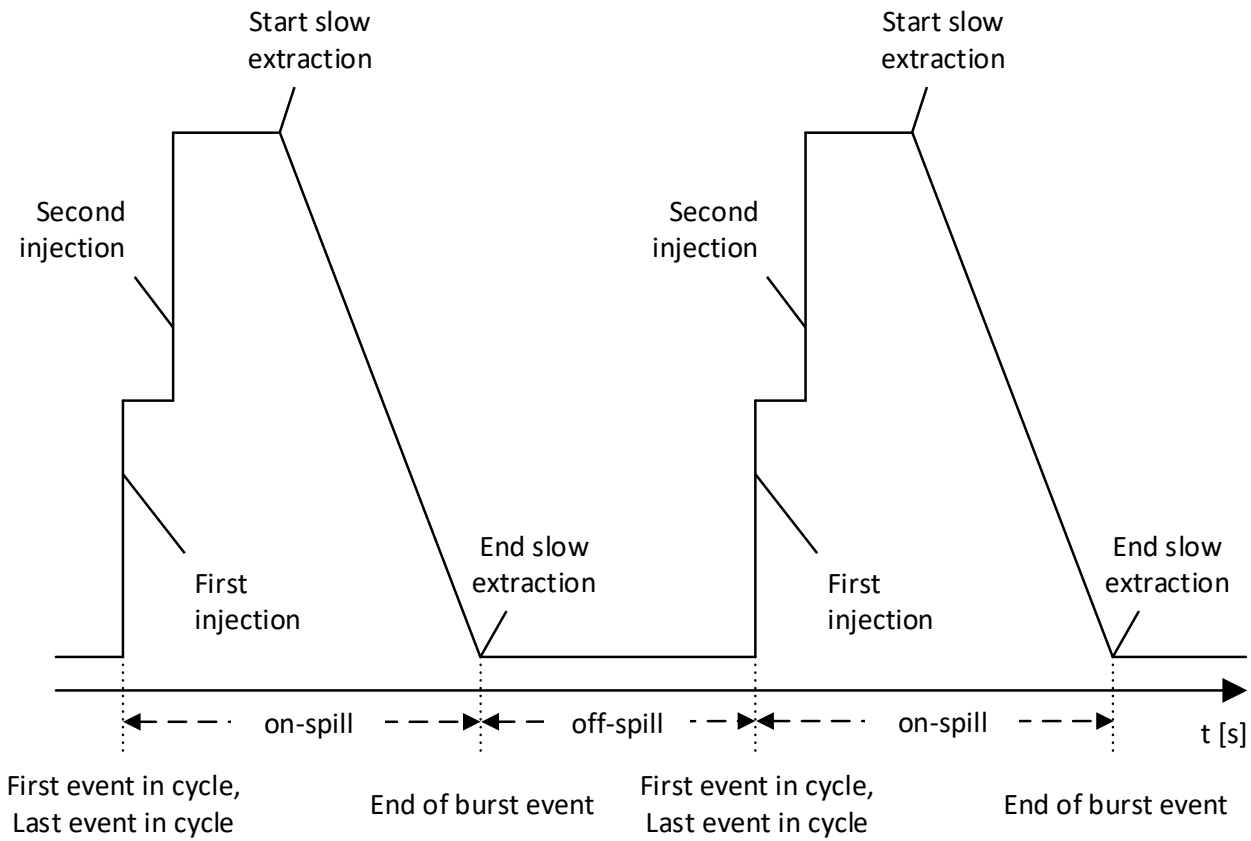
- Dry Run – mid-step before real data-taking
  - Verification – everything is prepared before real data-taking
  - Artificial run number and runs with 200 spills
  - It provides physics data to online monitoring tools
  - No data are stored in files
  - No records in the electronic logbook are created
  - Transition Dry Run/Run and vice versa is fast and smooth

Continuously Running Mode – Run

- Maximum number of spills (MNS) set in Runcontrol GUI
- Run – real data-taking
  - It is started from the active Dry Run
  - Real run number is incremented by one
  - It provides physics data to online monitoring tools
  - Data are stored in files
  - Records in the electronic logbook are created
  - Starting – reset the spill counter to 1 and move to Run
  - Terminating – move to Dry Run and reset spill counter to 1
  - The continuously running option $\rightarrow$ stay in Run or move to Dry Run after MNS

Proper Timing and Synchronization (T&S)

- SPS super cycle is a good candidate for T&S
- Actions are taken based on Dry Run/Run state
- First event in first spill $\rightarrow$ open/close files and create logbook records
- End of burst in last spill $\rightarrow$ reset spill counter and next run number
- Last event in last spill $\rightarrow$ fill logbook records and transition to Dry Run/Run



Conclusion

- DIALOG library provides efficient and reliable inter-process communication across different platforms
- DAQ Debugger creates crash reports and memory dumps whenever crash occurs
- iFDAQ UpTime is around 99.88% $\simeq$ 1 hour loss / month of data-taking
- Continuously running mode runs nonstop without any user intervention
- It helps to collect more physics data

Thank you for your attention