1. Introduction
Modern particle physics experiments are searching for more and more rare particles and decay channels. Therefore they have to operate with an increasing luminosity which produces huge data sets, where most of the events can be described by well understood physical processes and only a few are interesting for further analysis. To find these rare events is the challenge of the event selection and requires that most, if not all, of the data produced by the experiment is processed online with the same average rate with which the data are measured. To achieve the necessary computing power in an efficient way different hardware types from FPGAs, GPUs to multicore CPUs and mixtures of these systems are under study. To exploit parallel hardware different technologies are available. One very convenient approach for multicore CPU systems are message queues. Their application is shown here on the example of test beam data for the PANDA experiment[1].

2. PandaRoot, FairRoot and FairMQ
PandaRoot[2] is the Monte-Carlo simulation framework of PANDA. It allows in an easy way to simulate, digitize, reconstruct and analyse the complete physics program of PANDA with
Figure 1. Schematic drawing of the PANDA experiment at FAIR

its various sub-detectors. It is derived from the FairRoot framework[3], a common development at GSI for all FAIR experiments. One quite new feature of FairRoot is the implementation of FairMQ[4], an abstraction layer to give access to message queue systems like ZeroMQ[5] or nanomsg[6] which are both supported. The usage of message queues allows to pack the algorithms in small independent processes which are running in parallel and handle the inter-process communication via the message queues. Various different communication patterns are implemented in FairMQ to setup a topology most suitable for the processing needs. The two used patterns in this test are:

- push-pull: A one-to-n or n-to-one connection with guaranteed delivery of the data. If one of the pullers is not fast enough it slows down the whole data flow.
- pub-sub: A one-to-n connection which is non-blocking but with the possibility of data loss if the subscribers are not fast enough to handle the data.

3. The PANDA Experiment

The AntiProton-Annihilation at Darmstadt experiment (PANDA) at the Facility for Antiproton and Ion Research (FAIR), which is currently under construction, is an excellent example for future online data processing. It is a medium size particle physics experiment with the goal to study the behaviour of quantum chromodynamics in the energy range of charmonium. For this purpose it utilises a phase space cooled antiproton beam in a momentum range from 1.5 to 15 GeV/c which is incident on a fixed hydrogen or heavier nuclei target.

Once PANDA is running with its design luminosity of $2 \cdot 10^{32} cm^{-1}s^{-1}$ it will produce 200 GByte/s of raw data. Due to the very similar signatures of signal and background events all subdetectors continuously transmit their data to the online computing system without a hardware trigger. Each detector hit contains a time stamp which is the basis to combine the free running data into events. This data stream is then processed in an event selector which has to reduce the data by a factor of 1000 for permanent storage by selecting only those events which look physically interesting. After data selection PANDA will produce a total of about 1 PByte/year experimental data and in addition 2 PByte/year simulation data.

Fig. 1 shows a schematic drawing of the detector. The detector is subdivided into two parts, a central spectrometer using a 2 T solenoidal magnetic field and a forward spectrometer with a
2 Tm dipole magnetic field. Both spectrometers are equipped with a complete set of tracking, particle identification and energy measuring detectors. For the test of FairMQ components of the Micro Vertex Detector (MVD)[7] are used which is located directly around the interaction point. It comprises a silicon pixel detector in the inner part with about 10 million pixels and a silicon strip detector with about 200,000 strips in the outer part.

4. The Test Beam Setup
As a first test of FairMQ with real detector data a test beam of the MVD at the COSY accelerator in Jülich was chosen. Fig. 2 shows the setup. The first and the last detector layers both consisted of conventional silicon strip detectors for alignment purposes and was not part of this test. In between the two strip planes were four silicon pixel detectors with a continuous running ToPix4 ASIC. Each of the pixel detectors was read out via an attached FPGA board (XILINX ML605) which pushed the data via an UDP connection to two data taking PCs where the raw data for each ASIC was stored in an individual file. A 2.8 GeV/c proton beam was directed through the setup perpendicular to the detector planes.

5. Processing of Test Beam Data
The processing of the test beam data is subdivided into small individual executable programs which are connected via different types of FairMQ connections. The input to the processing stage is either the raw data of the beam time stored in binary files or directly the test beam data during the beam time. The output are fully processed tracks based on the data measured with the four pixel detectors.

The first stage of the processing is illustrated in Fig. 3. Four different types of executables exist:

- **File handling**: These programs read and write data to the hard disk:
  - *Raw Data Sampler* reads in the stored raw data of the test beam data. In an online operation this device will be replaced by a connection to the test beam DAQ.
  - *File Sink* writes the pre-processed hits of each ASIC into a root file.

- **Data Processing**: These programs modify the incoming data:
  - *Raw to Digi* analyses the measured raw data of 40 bit words and converts them into a *PndMVDTopixDigi* which contains information on which pixel fired, the deposited charge and the time stamp when this hit has happened.
Figure 3. Flow chart of the processing stages for one front-end ASIC

- **Sorter** sorts the incoming data according to its time stamp which is important to later combine the data from different front-end ASICs.
- **Digi to Hit** clusters fired digis if they are adjacent in space and time. This digi clusters are then transformed into a `PndMVDHit` by a charge weighted averaging. A hit contains the sum of the deposited charge, the mean time stamp and the 3D position of the hit in global coordinates. In addition all hits within a certain time window are grouped together into an event.

- **Data handling**: These programs distribute the data to different targets without modifying its content:
  - **Duplicator** takes the incoming data and sends it to several targets. For each target a fraction can be set if all the data is sent to it or only every \( n^{th} \) data object. This is used e.g. to send only every 1000\(^{th} \) digi to the online monitoring.
  - **Distributor** handles the data distribution to the parallel operating sorters. To ensure that each sorter gets the data within a certain time interval the distributor looks at the time stamp of the incoming data. If it is below an adjustable threshold the data is sent to the first sorter, if it is above it goes to the second sorter. If the incoming data is above an offset over the threshold the distributor increases the threshold and sends the data to the second and third sorter and so on.
  - **Merger** combines the data from the parallel operating sorters into a continuous, time sorted data stream.

- **Online monitoring** displays part of the data or aggregated values.

This first processing stage exists for each ASIC as can be seen in Fig. 4. Two additional programs then operate on the total data-set of the four ASICs:

- **Event Builder**: It receives the hits of each ASIC in four parallel data streams. Each data stream is time sorted and the hits are grouped into events. To combine the four streams into one the Event Builder takes the first event of all streams, checks if the event with the lowest time stamp has other events within a certain time window. If this is the case the events are combined and read out. The data of the read out events is replaced by new data from the specific data stream and the process continues until all data is processed.
• **Track Finder/Fitter.** It takes the combined events and does a straight line track finding and fitting of the data.

Not shown in the diagrams is a global controller program which gives a common start signal to the four parallel running *Raw Data Sampler* to ensure that the data processing really happens in parallel.

All FairMQ connections between the different programs are of type push-pull to ensure that no data is lost. The only exception are the connections to the online monitoring and the global controller. Here a pub-sub scheme is used to prevent that an online process is blocking the data transmission.

6. **Online Monitoring**

Each important step in the processing chain also generates monitoring data which is sent via FairMQ to a GUI which visualizes the data for the test beam operator. The GUI is written in Qt and uses in addition the histogram classes of ROOT, the FairMQ part of FairRoot and the data classes of PandaRoot. Fig. 5 shows the two most important screens of the online monitoring program. On the first page you can see the four 2D histograms which show the hit distribution on the four different ASICs. These histograms get every 1000\(^{th}\) event of the *Raw to Digi* converter via the *Duplicator*. The four smaller 1D histograms show the hit multiplicity inside a cluster of digis of the complete data. This numbers are collected in the *Digi to Hit* converter and send with a lower rate to the GUI.

On the second screen the combined data of all four ASICs is shown. The lower left 1D histogram with the title ”Sensors per event” shows how many ASICs have hits inside one event independent on how many hits a single ASIC has. Those events with 4 ASICs fired are then given to the track finding and -fitting from which the results are shown in the remaining 3 histograms. The top 2 histograms show the origin and the direction of the fitted tracks while the lower right histograms shows how many tracks have been found in one event.
Figure 5. Two screenshots of important information from the online monitoring program showing live data from the ongoing data taking from all stages of the processing.

7. Performance
The performance in a distributed system was tested with 4+1 PCs. The first four Linux PCs were each processing the data of one ASIC. One of them also did the event building and track finding/-fitting. These four PCs were connected via Gbit ethernet. The fifth MAC PC was used to run the online monitoring and the global controller to start the processing. This one was connected with the other PCs via a 100 Mbit ethernet connection.

The processed data corresponded to one beam spill with a length of 20 seconds. Each ASIC collected about 55 MByte of data during this time. The subdivision of the processing into single executables allowed to used all 8 cores of the Linux machines for 100 % during the run time. The individual processing stages for each ASIC took 15 seconds for full processing which is below the 20 seconds it took to create the data, which was the requirement. The processing time nevertheless increased to one minute once the event building with all four ASICs was included. The later track finding and fitting did not have any influence on the processing time.

The reason for the strong increase of the processing time is still under study.

8. Summary and Outlook
A first test of FairMQ was made for the PANDA experiment. A very simple setup with just four pixel detectors was chosen with the requirement to process the data with the same speed it was produced. In total 50 processes are dealing with the data running on four different PCs with an additional control PC. The change from the standard implementation in PandaRoot towards a system based on FairMQ proved to be easy and reliable. With this setup it is possible to process the test beam data up to fully reconstructed tracks. The achieved performance for the individual processing stages of the four detectors was faster than the data production in the test beam. However, the performance dropped significantly when the data from all four detectors was combined into a common event structured data stream, which is subject to further development.

The presented setup will be used for the next MVD beam time end of May 2017.

Bibliography
[1] PANDA Collaboration, W. Erni, I. Keshelashvili, B. Krusche, M. Steinacher, Y. Heng, Z. Liu, H. Liu, X. Shen, O. Wang, H. Xu, et al. Physics Performance Report for PANDA: Strong Interaction Studies with Antiprotons. March 2009.
[2] Stefano Spataro. Event Reconstruction in the PandaRoot framework. Journal of Physics: Conference Series, 396(2):022048, 2012.

[3] M. Al-Turany, D. Bertini, R. Karabowicz, D. Kresan, P. Malzacher, T. Stockmanns, and F. Uhlig. The FairRoot framework. Journal of Physics: Conference Series, 396(2):022001, 2012.

[4] M. Al-Turany, D. Klein, A. Manafov, A. Rybalchenko, and F. Uhlig. Extending the fairroot framework to allow for simulation and reconstruction of free streaming data. Journal of Physics: Conference Series, 513(2):022001, 2014.

[5] http://www.zeromq.org/.

[6] http://nanomsg.org/.

[7] The PANDA Collaboration. Technical design report for the: Panda micro vertex detector. Technical report, Aug 2012.