NA62 Dataflow

DAQ→CASTOR

Marco Boretto EP-DT-DI CERN

NA62 experiment

The NA62 experiment is located at CERN Super Proton Synchrotron (SPS) and studies ultra-rare kaon decays.

I will focus on the last part of the Data Acquisition (DAQ) chain that consists of copying the data from the DAQ storage buffer to CASTOR.

Spills and runs

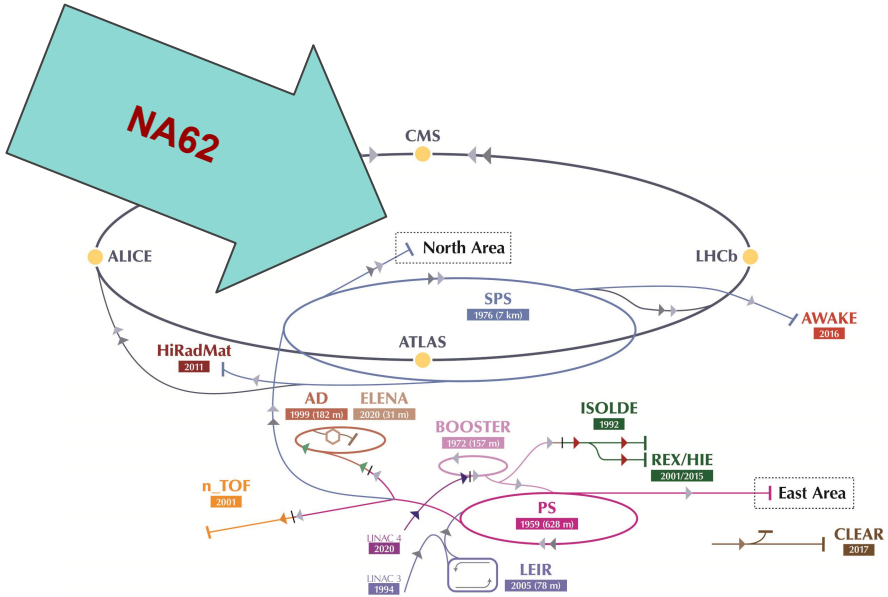
The DAQ system follow the SPS spill periodicity:

~4 s spill + ~18 s interval

Spill (or burst): smallest amount of data that you can collect

Run: group of spills took in the same conditions (~1500 spills)

Sample: group of runs



Kinds of data

Raw data

The data produced by the detector.

- **Producer hosts:** na62merger1, na62merger2 and na62merger3
- **Files produced**: 1 files per spill, all of them must be transferred to CASTOR.
- **Average size:** ~3-4 Gbyte per file

Primitive data

Used to verify the functioning of the trigger system.

- **Producer host:** na62primitive
- **Files produced**: 5 files per spill. Only a fraction of them is copied on CASTOR (1/10)
- **Average size:** ~20 Mbyte per file

The data don’t need further processing and have to be stored permanently as they are.

The NA62 CDR

CDR stands for **Central Data Recording**, it was the name of the legacy software that NA62 was using to transfer the data.

In 2017 I dropped the old code (too old) and I developed a new one.

CDR software developed by me with the help of the CERN IT-ST team.

- successfully used for 2 data-taking period: 2017 and 2018
- Transfer the data to CASTOR over the grid thanks to the introduction of FTS

CDR infrastructure

- The storage element Globus-gridftp-server deployed on all the hosts: na62merger1-2-3 and na62primitive
- MariaDB database, to book-keep the files/transfers and their status
- Master and slave MariaDB instances hosted on na62merger2 and na62merger3

The FTS proxy:
https://fts3-daq.cern.ch:8446

- hosted on a Virtual Machine
- deployed and managed by IT

CDR code

Book-keeps the files produced and orchestrates the file transfers from the DAQ to CASTOR

Written in Python 2

External Python module:

- **fts3** allows to communicate with the FTS proxy and control the storage element
- **gfal2** library tape status check

Files routing

DAQ files are organised in subdirectories on a run base

The CASTOR destination path is evaluated from the DAQ file name.

File name:

```
na62raw_1563877735-01-009548-4407.dat
na62raw_<timestamp>-<merger_id>-<run_id>-<spill_id>.dat
```

Castor path:

```
base_path/009548/na62raw_1563877735-01-009548-4407.dat
```

Files deletion policy

The CDR reschedules the failed transfers until they are successful.

A file is deleted **from the CDR** on the submission host if:

1. **The file is successfully migrated on tape**
2. 24 hours has passed since the file creation

FTS does not allow to check whether a file has been migrated or not on tape.

I use a wrapper around the GFAL2 library.

I know that there is some work on FTS to allows tape checks.

CDR development

Development of a “generic” version of the CDR code that can be used in other experiments.

Done by me and Cristina a student supervised by Edward Karavakis.

Development (imprecise) todo list:

- Allows parameters customization (almost done)
- Support for Castor and CTA
- CI tests and documentation
- Make the git repository public
- Remove the GFAL dependency
- Migration to Python3 (blocked no FTS module)

CTA migration go/no go?

I wish to have a CDR software that can work transparently with CASTOR and CTA in order to be prepared for any eventuality.

According to Edward, to support CTA in the CDR we "just need to change the destination endpoint".

I'm looking forward to test it.

- When the small experiments will be migrated?
- Is not clear how the data transfer to CTA will take place:
  - DAQ -> CTA (as it is now)
  - DAQ -> CTA and DAQ->EOS (double up the data flow from the DAQ)

DAQ -> EOS -> CTA **will complicate unnecessarily the DAQ flow**

Conclusions

Other thoughts

For some time after a successful transfer the file coexist on tape and disk.

The files on disk are used for the “**Prompt reprocessing**” a procedure to assess the data quality.

Dry-run from 10 to 20 August

- I will test the CDR code that allows customization

Do you have an CTA public endpoint that we can exploit for some tests?

The database schema

- One table: **transfers**
- Transfer re-submission mechanism incorporated in the table
- **resubmit_id** foreign key constraint to the new transfer (a file can be re-submitted infinite time)
- **src_path** column de-normalised (design choice)
- **src_host** column ENUM field: na62merger1-2-3, na62primitive (**the submitting host names should not be hardcoded in the table**)

id	src_host	src_path	transfer_id	status	resubmit_id
30	merger1	file30.dat	87gyht	transferred	null
31	merger2	file31.dat	86yrf8	failed	32
32	merger1	file31.dat	33etjln3	failed	34
33	merger3	file32.dat	vjh589	transferring	null
34	merger1	file31.dat	gjuy54	transferred	null

New file trigger

The DAQ produce files in a directory, this process can last up to ~10 seconds.

The CDR reads the files in the very same directory.

How avoid to transfer files while they are still in the writing process?

The DAQ changes the owner and the group at the end of the writing process.

The new file trigger should be customizable

- looking for file extensions
- new file in the directory

```
-rw-r--r--  1 root root  456 mag 31 18:00 file1.dat (not ready)

-rw-r--r--  1 na62cdr na62cdr  748 mag 31 18:00 file1.dat (ready)
```

Hosts configuration and code deploy

The software is deployed on 4 server of the NA62 DAQ system.

The preparation of the environment is automated with a bunch of Ansible playbook, not provided with the code.

Ansible automates:

- globus-ftp-server installation and set up
- MariaDB master/slave replica installation and configuration
- deploy of the CDR code (git clone)