

CERN Summer Student Program Project

CDR Update for NA62

Summer student name: Ting Yin

Supervisor: Marco Boretto, Giovanna Lehmann Miotto

24/6/2019 - 16/8/2019

CERN, Geneva

Abstract

During this program I learned about the current NA62's CDR system and the technology required. I worked on the upgrade of the database schema and related components that contribute to the CDR's continued compliance with the CASTOR storage system. NA62 is located at CERN SPS, its purpose is to precisely measure rare kaon decays. The data produced from the detector must be transferred to CASTOR, a storage system developed at CERN for archiving very large data volumes. The CDR software tracks the files produced into a relational database and schedules the transfer to CASTOR. As a result of the CASTOR upgrade foreseen for 2021, the CDR software needs to be refurbished to support new storage types.

CDR update

As shown in figure1, the current CDR system is designed to transfer the raw data to CASTOR, CERN Advanced STORage manager (CASTOR), a hierarchical storage management system which was developed at CERN for archiving physics data. The CDR system exploits File Transfer Service (FTS), the Data Transfer Service used at CERN which provides data movement service aiming to reliably copy one Storage URL to another.

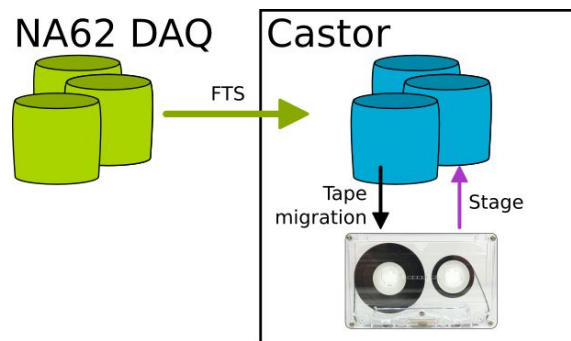


Figure1: schematic layout of the NA62 data transfer system

To support the data flow change shown from figure1 to figure2, we need to redesign the database schema, and update the controller components to work with the new schema in order to achieve the new data transfer capability.

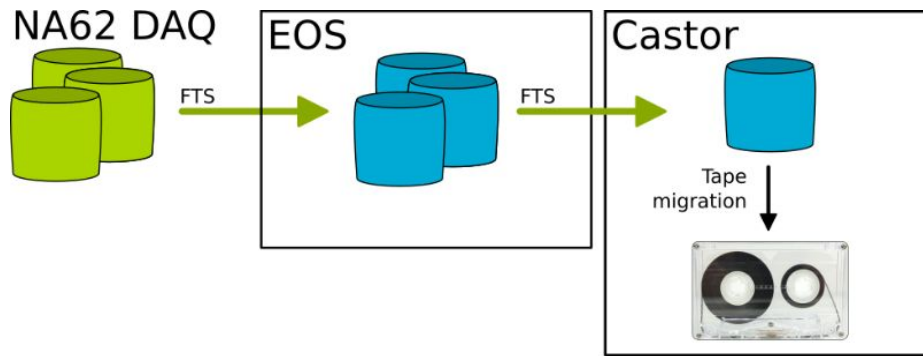


Figure2: schematic layout of the NA62 data transfer system upgrade

The CDR is implemented with python, alembic, SQLAlchemy, MariaDB. We use alembic and SQLAlchemy instead of raw SQL because SQLAlchemy is a Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL, and alembic is a lightweight database migration tool for usage with the SQLAlchemy Database Toolkit for Python. After I familiarized myself with these tools, I made an example database project on my dedicated virtual machine. The demo project can be found here <https://github.com/emmayint/bookorder>. For IDE I chose Visual Studio Code Remote on CentOS7 x86_64 instance.

Database schema upgrade with alembic:

After examining the unidirectional data migration process from merger to CASTOR, we began designing new database schema to better suit the new data transfer between mergers, CASTOR and EOS (a disk-based, low-latency storage service, having a highly-scalable hierarchical namespace, and with data access possible by the XROOT protocol).

The old schema has one table “transfers” that contains all the information about file tracking, bookkeeping and job submission. After careful consideration and discussions, we decided on a new design composed of three tables “files”, “hosts” and “transfers” that separate file bookkeeping, hosts information and transfer job with foreign key constraints as shown in figure3. The new database schema is instantiated with alembic upgrade and initial rows and values are inserted.

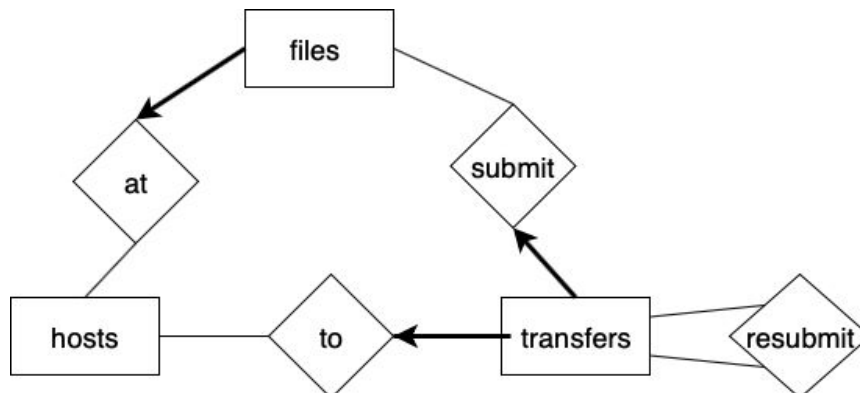


Figure3: CDR Entity Relationship Diagram

Database schema composed of a table “transfers”:

transfers		
'id'	BIGINT	primary_key
'file_name'	String(255)	
'src_host'	Enum	
'src_path'	String(255)	
'dst_path'	String(255)	
'status'	Enum	
'job_id'	String(255)	
'created_at'	TIMESTAM P	
'updated_at'	TIMESTAM P	
'transferred_at'	TIMESTAM P	
'file_size'	BIGINT	
'is_src_deleted'	Boolean	
'deleted_at'	TIMESTAM P	
'transfer_attemp '	Integer	
'resubmit_id'	BIGINT	Foreignkey
'is_invalid'	Boolean	
'comment'	String(255)	
'is_on_tape'	Boolean	
'on_tape_at'	TIMESTAM P	

Updated database schema composed of tables “files”, “transfers”, and “hosts”:

files			transfers		
'id'	BIGINT	primary_key	'id'	primary_key	increment
'file_name'	String(255)		'file_id'	BIGINT	Foreignkey
'created_at'	TIMESTAMP		'job_id'	Integer	
'updated_at'	TIMESTAMP		'dst_host_id'	BIGINT	Foreignkey
'file_size'	BIGINT		'dst_path'	String(255)	
'src_host_id'	BIGINT	Foreignkey	'status'	Enum	

src_path'	String(255)		transferred_at'	TIMESTAMP	
is_src_deleted'	Boolean		transfer_attemp'	Integer	
deleted_at'	TIMESTAMP		resubmit_id'	BIGINT	Foreignkey
			is_invalid'	Boolean	
			comment'	String(255)	
hosts			is_on_tape'	Boolean	
id'	BIGINT	primary_key	on_tape_at'	TIMESTAMP	
name'	String(255)		created_at'	TIMESTAMP	
host_url'	String(255)		updated_at'	TIMESTAMP	

Models update (ORM):

New models of class “File”, “Host”, and “Transfer” were created based on the new database schema Object-Relational Mapping with SQLAlchemy.

Controllers update:

The updated controller files should work with the new database schema, providing better modularization and isolation for file tracking, book-keeping and file transfer. For my final week, I will be modifying controller components to fit the new database schema, modularized file bookkeeping, and job scheduling and submission. All of the work I have done on updating the CDR can be found in the gitlab dev branch <https://gitlab.cern.ch/ep-dt-di/daq/na62/cdr/tree/dev>.

FTS update:

FTS set up and completing the fts-installation documentation in gitlab.

Future work

The future work for CDR system should focus on controller files update, especially CdrController.py, SourceController.py, so that they can work with the new database schema, and provide better modularization and isolation for file tracking, bookkeeping, and file transfer.

More resources are needed to incorporate EOS into the system.

References

“The Data acquisition system of the NA62 experiment at CERN”, Marco Boretto, on behalf of the NA62 Collaboration

“Introducing CERN Tape Archive (CTA)”, Marco Boretto, Eva Barbara Holzer

“Integrated Trigger and Data Acquisition system for the NA62 experiment at CERN” by B. Angeluccia, C. Avanzinia, G. Collazuol,b, S. Galeottic, E. Imbergamod, G. Lamannab, G. Magazzu`c, G. Ruggierob, M. Sozzia, S. Vendittia, on behalf of the NA62 Collaboration*

<https://www.sqlalchemy.org>