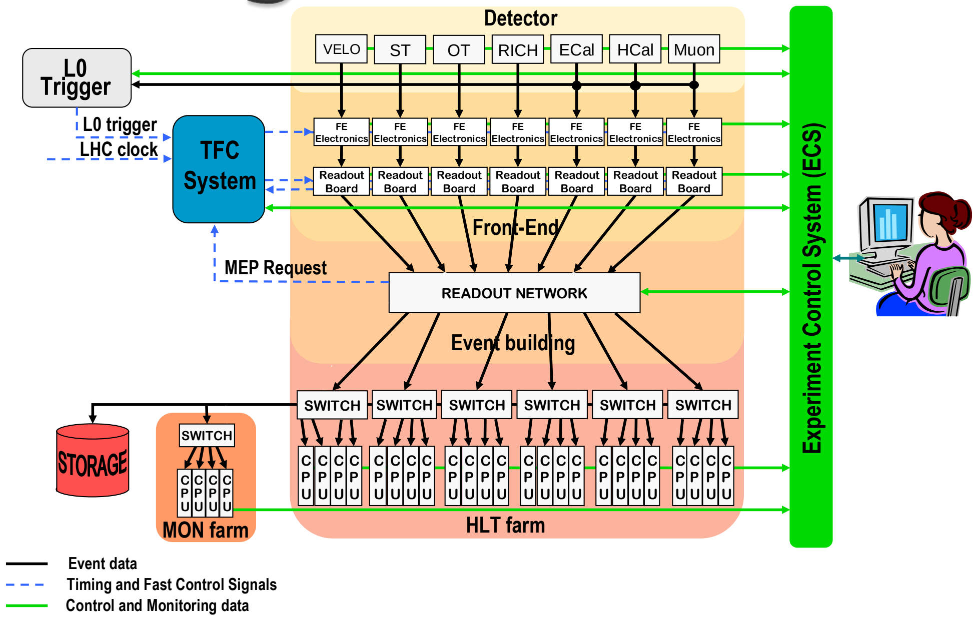
DAQ Systems configuration

ALICE, ATLAS, CMS & LHCb joint workshop on *DAQ@LHC*

12.03.2012

DAQ Systems Configuration

- A lot of subsystems need to be configured for DAQ
  - Front-end electronics
  - Readout Units
  - Trigger
  - Event Builder
  - HLT, …
- DAQ has to run with different configurations for several types of run
  - Calibration runs, technical runs, Cosmics, …
  - Sub-detector stand-alone runs
  - Physics



- There needs to be an easy, reliable and fast way to configure the different subsystems according to the different run types

DAQ Systems Configuration

- Each Experiment implemented the DAQ systems differently
  - Different technologies
    - Web based
    - PVSS based, …
- Each Experiment implemented DAQ configuration differently
  - XML Descriptions
  - Named datasets in DB
  - Object Persistent DB

ALICE

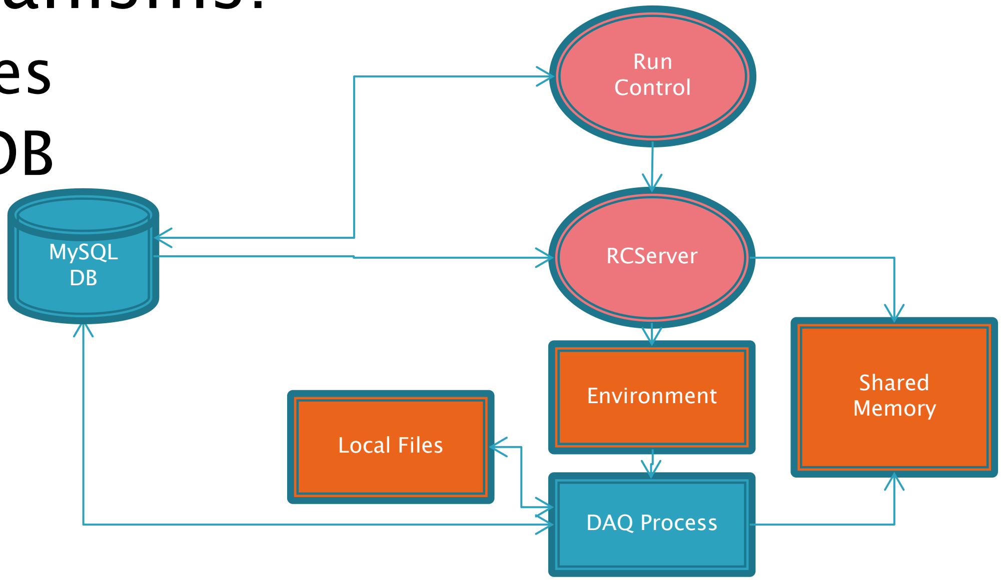
- Architecture summary:
  - Global DAQ configurations stored under a Name
  - Specific machine configurations stored under a Role Name
  - Configurations are stored in a MySQL Database
  - Configurations are retrieved via C SQL API
- Databases
  - MySQL DB
    - More static DAQ configuration settings defined by Name
      - Streams
      - Timeouts, …
    - Configurations for each machine defined by Role Name (e.g. “LDC-TPC-1”)
- The MySQL DB is populated by system experts
  - Via Run Control Human Interface
  - Via a tool developed to insert configurations – editDb

ALICE

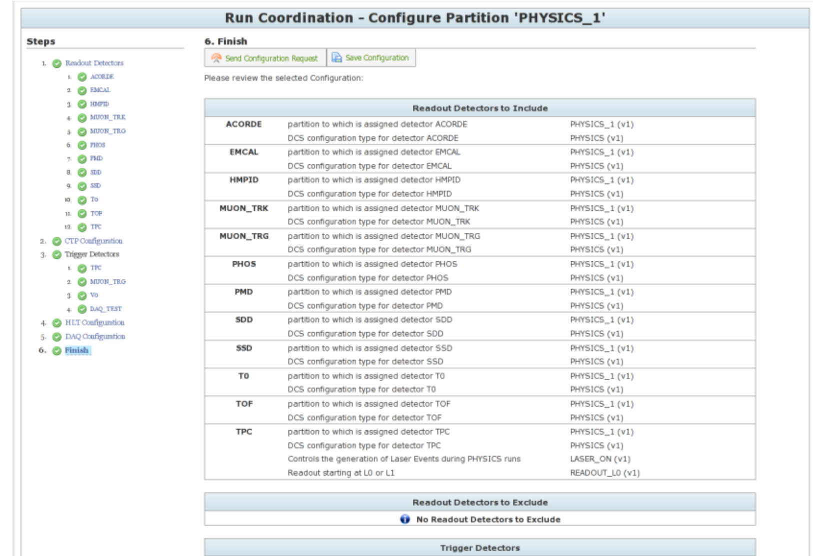
- The ALICE Run Control is based on a central Run Control and distributed Run Control Servers (RCServer)
  - The RCServers are responsible for starting and configuring the DAQ processes
  - An RCServer runs on each machine and has a Role Name defined
- Upon start of the Run Control, a DAQ configuration, stored with the name "DEFAULT", is loaded from the database
- Another configuration can be selected and loaded from the database, by configuration Name
- Before starting a run, other parameter settings can be performed. The new parameters are not saved automatically in the DB.
- The configuration is retrieved from the database by the RCServers
  - The configuration is loaded into shared memory which will be accessed by the processes started by the RCServers

ALICE

- The RCServer processes communicate with the central Run Control via SMI++
  - Certain configuration parameters are transmitted via SMI++ parameters
- The DAQ processes configuration is also done via other mechanisms:
  - Local configuration files
  - Access to the MySQL DB
    - Configurations in the DB are accessed by Role Name



ALICE



- Trigger configurations are stored on another MySQL DB (ACT DB)
  - A tool “ALICE Configuration Tool” (ACT) was developed to manage the configurations on the DB
  - At the start of Run, the Trigger system downloads the active configuration
- HLT configuration:
  - On the ECS (Experiment Control System) some parameters are sent to the HLT System
    - HLT mode
    - List of active links
    - List of Trigger Classes
  - The configuration is then handled by the HLT system
- Front-end configuration is done via the DCS
  - Using PVSS

ATLAS

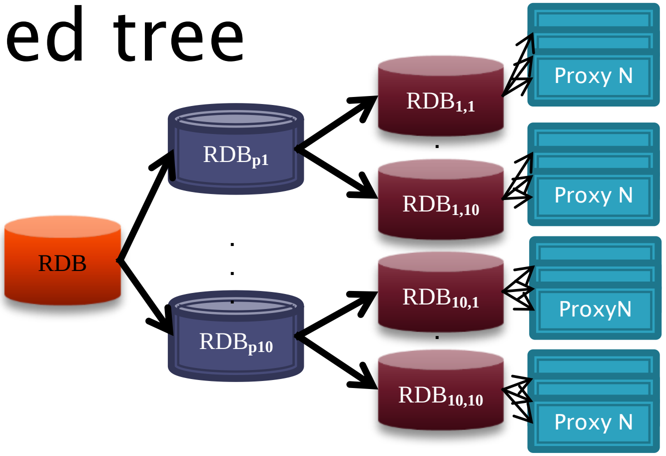
- Architecture summary
  - Based on object approach
    - Configuration is a graph of linked objects
  - Classes and objects are stored as XML files
    - XML Schema files – define object classes
    - XML Data files – store database objects
  - XML information is stored in a protected file repository and archived on a relational database
  - Home made object database is used to access XML information – OKS
- Databases
  - OKS DB provides overall DAQ system description for Control, Monitoring and Data flow
  - Trigger, HLT and detector configurations are partially covered by OKS

ATLAS

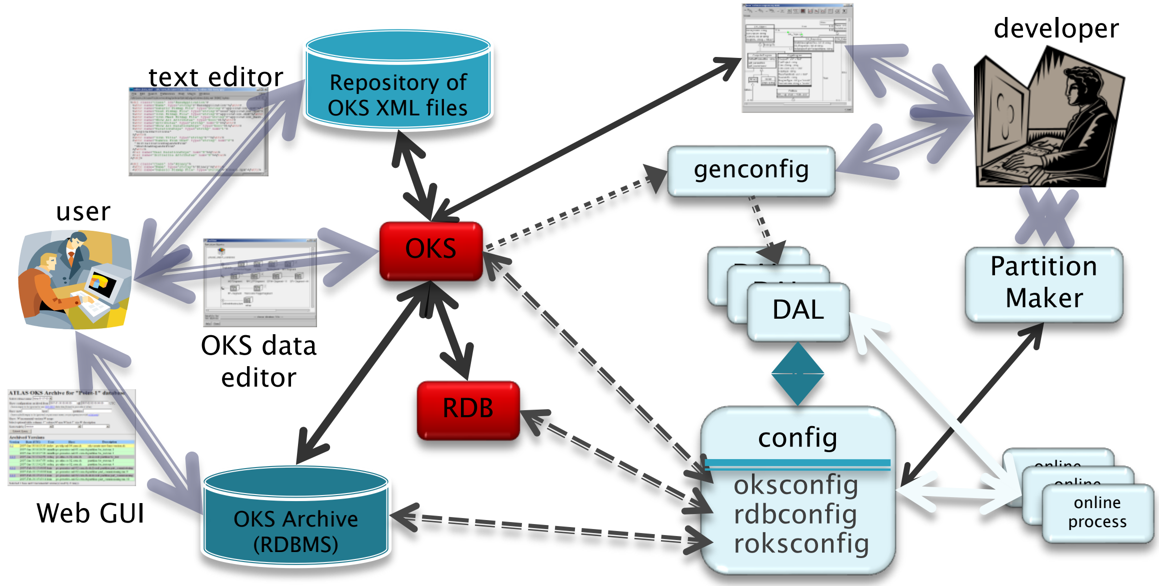
- Configuration objects structure is defined by the database schema
  - Common database schema was agreed with trigger and detector groups
  - Groups can extend the schema to introduce properties for their configuration objects
    - OKS classes support inheritance with polymorphism
- The OKS database is populated by the relevant experts
  - GUIs are available to create XML schemas and data
  - A tool is available to generate automatic configurations with minimal user input
    - Partition Maker (Python based and with Python interface)

ATLAS

- Access to the OKS database is provided via Remote Database servers (RDB)
  - Provides access to the OKS Db from different clients without a common file system
  - Caches results of OKS queries
- RDB is developed on top of CORBA
- To address scalability requirements RDB servers are setup as a balanced tree
- Current deployment of the RDB Servers
  - XML repository -> RDB Master -> Pool of RDBs -> RDBs running on the Racks -> RDB Proxy running on the nodes



ATLAS



- ATLAS applications do not use the OKS or RDB APIs directly
  - Avoid dependencies on DB implementations
  - 2 layers
    - Configs – abstract interface to work with DBs and the configuration objects
      - Available as plugins:
        - OKS XML Files
        - OKS Relational archives
        - RDB Server
    - Data Access Libraries (DALs) – uses configs to map database schema on the relevant language class and instantiates their configuration objects from DB data
- Data Access Libraries are automatically generated from the OKS schema for C++, Python and Java

ATLAS

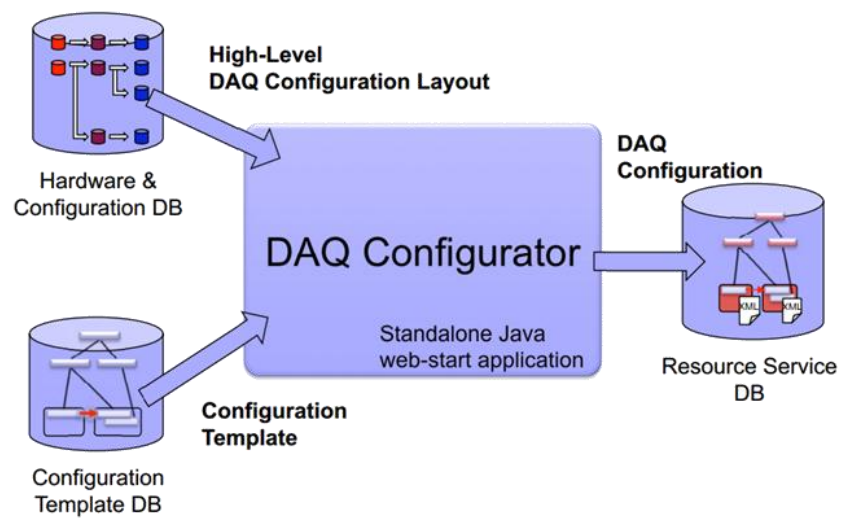
- Only one configuration is loaded at a given time in the OKS DB
- On start, Run Control reads information from OKS
  - Applications to start
  - Where should they be started
  - Application parameters
  - Each application loads the appropriate config plugin
- On the configure transition
  - Applications instantiate their configuration objects using the corresponding DAL

CMS

- Architecture summary
  - Configurations are stored in a relational schema
    - As XML Descriptions for DAQ software processes
  - Configurations are stored in ORACLE DB
  - Run Control queries the DB via JDBC
  - XML configurations are passed to the online software processes via SOAP
- Databases
  - Resource Service DB
    - Stores all the information for the dynamic configuration of the hosts
    - DAQ Configurations based on the templates from other databases
    - Stores all the parameterized XML descriptions to configure XDAQ executives
  - Configuration Template DB
    - Stores slowly changing information templates (e.g. composition of Functional Units)
  - DAQ Hardware and Configuration DB
    - Stores frequently changed information
      - High level configuration layout (e.g. location, mutiplicity and connectivity of Functional Units)

CMS

- How to populate the Resource Service DB
  - DAQ Configurator tool
    - Reads configurations templates and high level layout from the Configuration Template DB and DAQ Hardware and Configuration DB
    - Computes and sets the parameters from the templates
    - Allows ad-hoc user input
    - Creates XDAQ Executives XML configuration documents
  - Other GUIs
- Configurations in the Resource Service DB
  - Versioned
  - Tree like structure
  - Used by Run Control and all sub-systems

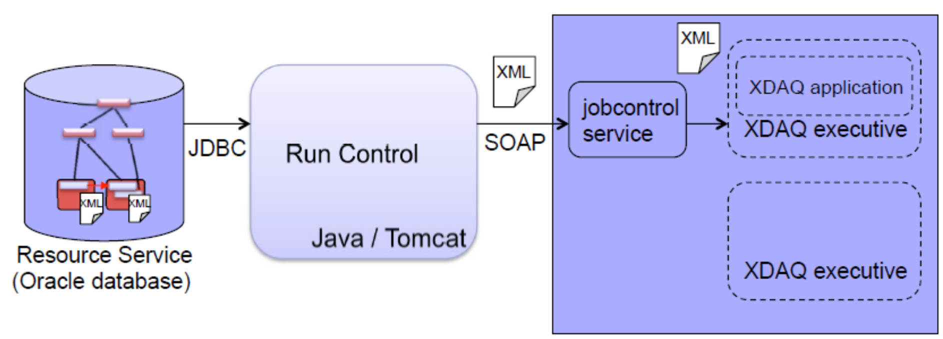


CMS

- DAQ online software is based on the XDAQ framework
  - XDAQ Executives (processes)
    - e.g. Builder Units, Filter Units, Event Managers, …
  - XDAQ Applications
    - One or more per Executive
    - Perform actions
      - Setting front-end parameters
      - …
  - XDAQ Executives are highly configurable through XML Documents
    - Determines the role of the executive
    - Sets up software environment
    - Contains applications and parameters to be loaded by the executive
    - Determines collaborating applications

CMS

- At the start of a session, the Run control configures the DAQ cluster dynamically
  - The currently registered configuration gets loaded from the Resource Service DB
    - Holds the info about the Function Managers to be loaded
    - Holds hierarchical information for the FMs
- The started Function Managers (FMs) load all the XDAQ executives according to the configuration currently registered for each sub-system
  - Job Control Service (XDAQ Executive) runs on all hosts in the cluster
  - Job Control Service reads the XML Document from the configuration
  - XML Documents contain the information about XDAQ executives and applications to be started on the hosts
- XDAQ configuration determines:
  - Configuration of custom hardware
  - Configuration of Super-Fragment Builder
  - Event data flow topology in Event Builders



CMS

- Trigger and HLT configuration
  - Trigger and HLT configuration can be selected on the Run control
  - Default configurations are registered in the FMs
  - The Function Managers load the different configuration settings for the Trigger and HLT accordingly
  - Configurations for Trigger and HLT are stored in different databases
  - Configurations can be changed at runtime, which triggers FM reloading requests
  - Configuration changes in the DBs will be detected by Run Control immediately

LHCb

- Architecture summary
  - Configurations are stored as named Recipes
    - Recipe types define for each configurable device type which settings are to be stored in a Recipe
    - Recipes implement the Recipe types with the valued parameters
  - Configurations are stored in ORACLE DB and in PVSS Cache
  - Configurations are accessed by name from PVSS, via an RDB Manager
- The Recipe names follow a convention
  - Hierarchy of activity type (e.g. “PHYSICS|pA|VdM”)

LHCb

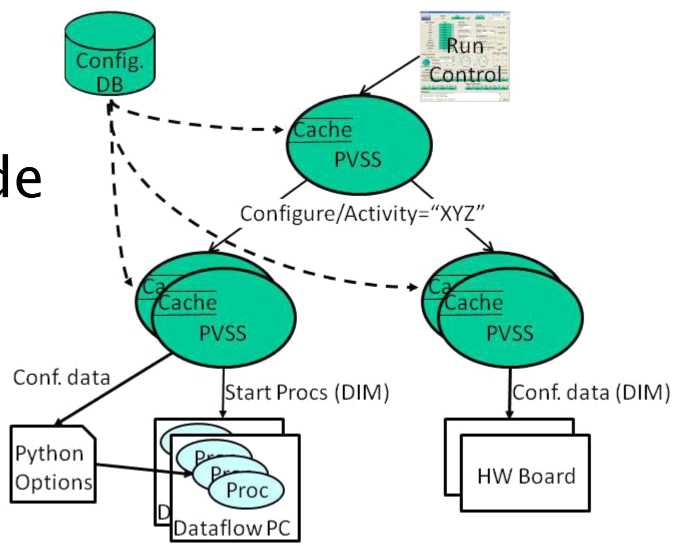
- Databases
  - All the configurations are stored in an Oracle DB – Configuration DB
- Configuration DB
  - Is populated mainly outside running periods by the relevant experts
  - Is populated using PVSS tools
- Cache
  - Each PVSS project where a configuration will be used can have a cache with the relevant Recipes

LHCb

- Run Control is based on the PVSS Scada software
- PVSS Run Control integrates all the systems:
  - Trigger
  - Front-end hardware control
  - Readout Supervisors
  - HLT
  - Event Builder control
- Configuration of all the sub-systems is treated in a similar manner
- The Recipes are applied according to the currently set activity in the Run Control
  - Recipes to be loaded are matched by name to the set activity
  - Recipes loaded follow the activity type hierarchy
    - e.g. If the set activity name is "PHYSICS|pA|VdM", the devices to be configured will check if a recipe that matches the name and apply it, if there isn't, they'll check if there's a "PHYSICS|pA", if there isn't they'll proceed to check for one named "PHYSICS"
  - "Default" named Recipes for non Activity dependent configurations.
    - In case no match is found for the given activity, default settings will be applied

LHCb

- Dataflow Processes configuration
  - Based on Offline Gaudi FW
  - Are configured via job options files (python)
  - There are static job options (changed with new releases of the Online SW)
  - There are job options files dynamically created from the run control
    - According to the set Activity
    - According to the Partitioning Mode



LHCb

- Trigger configuration
  - Uses an additional Trigger Configuration Key (TCK)
  - The Recipe loaded for current Activity contains the TCK of the HLT Recipe to be loaded
  - Can be changed at Runtime without major reconfiguration

Conclusions

- Different philosophies and technologies for DAQ implementations dictated different approaches for the system configuration
- Some similarities
  - The experiments implemented a 2 step approach
    - Define the schema of configurations
    - Define the parametrized configurations
  - ATLAS and CMS implemented XML files to store configuration data
  - LHCb and ALICE store the configurations as named sets of parameters
  - Usage of dbs is pervasive
    - Configuration values direct storage
    - XML files storage
    - Configuration objects storage
  - Trigger handled slightly differently
    - Change trigger settings without whole DAQ system reconfiguration

Acknowledgements

- A thank you to Clara Gaspar, Hannes Sakulin, Igor Soloviev and Vasco Barroso for the information provided and patience explaining it.