Technical Runs: Partition configuration

Argyris Zardilis

CERN

DAQ/HLT meeting, 15 August 2013

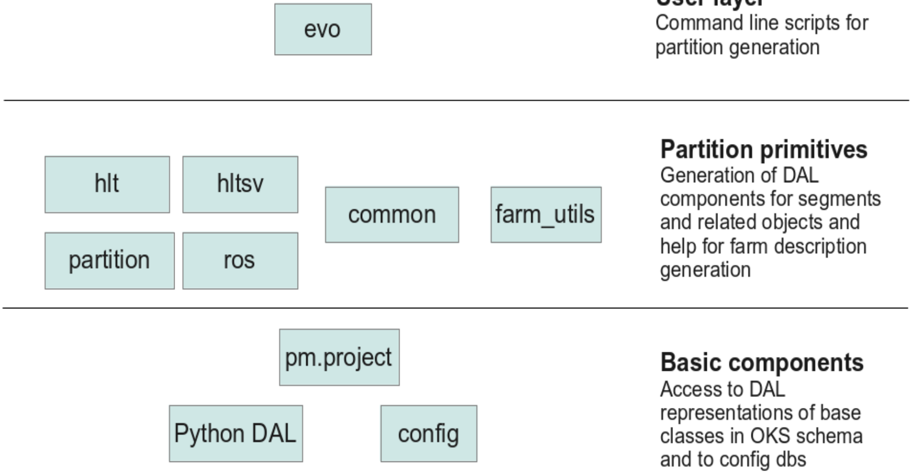
Motivation

- Large amount of information in a partition, hard to create its configuration manually
  Solution: Use a tool that automates all or part of the process
- *PartitionMaker*: fully featured configuration generation
  - Depends heavily on design and schema
  - Difficult to keep up to date with constant schema changes during design evolution
- Last Technical Run: need to create partition of different flavours to test different components of the system and their combinations.
- Manual partition configuration generation laborious and time-consuming
- Minimise time loss during TR

Solution

- Created a command-line script that automates generation process
- Started as a temporary solution for TR2 but since grew to a more complete and configurable command-line tool akin to *PartitionMaker*.
- By no means as complete or comprehensive but covers basic use cases
- Build on top of config and Python DAL package to get access to base classes defined in OKS schema
- Also uses pm.project from *PartitionMaker* to get a convenient handle to the config db

Solution schematically

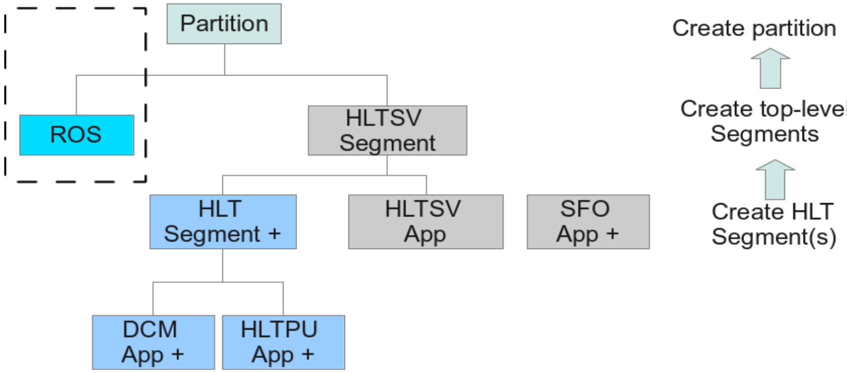


User layer
Command line scripts for partition generation

Partition primitives
Generation of DAL components for segments and related objects and help for farm description generation

Basic components
Access to DAL representations of base classes in OKS schema and to config dbs

Partition generation flow



Bottom-up approach. Start from primitive segments(don't contain other segments) and then pass their handle to their container segments and so on.

Capabilities

- Create localhost or multihost partitions in testbed/P1
  - farm description loaded from user provided python dictionary (a la *PartitionMaker*)
- Create standard partitions: only DCMs, only HLTPUs
  - customisable through command line parameters
- Create standalone, pluggable HLTSV segment
- Doesn't handle ROS segment generation
  - Not needed during TR (standard ROS segment)
- Also includes configuration for standard monitoring applications
  - Histogram/IS Gatherer
- Other configurable parameters: extra includes, data networks, repository root

Example usage

Create a DCM only partition with a provided python file 'farm_gen' containing farm description with partition name 'az_test' and repository root my home directory:

- `tdaq_python pm_evo.py -p az_test -f farm_gen --dcm-only -r tbed/user/azardili/installed`

Create a DCM/HLTPU partition with a provided python file 'farm_gen' containing farm description with partition name 'az_test' with PuDummy.data.xml as extra include:

- `tdaq_python pm_evo.py -p az_test -f farm_gen -I PuDummy.data.xml`

Conclusions

and Future Work

- A crude version proved useful for some use-cases in TR2.
- Short term plans: Use new polished version more extensively in next TR or for tests in testbed
  - Bonus: can be used by anyone now and it doesn't require knowledge of the code anymore!
- Use feedback from usage experience for subsequent development/evolution of *PartitionMaker*
  - Use-cases for new DF/run control/monitoring
  - New requirements
- Current version lives in a private git repo on my public afs partition which you can clone:
  - `/afs/cern.ch/user/a/azardili/public/partition_maker`