TRIUMF

Saved settings

Ben Smith

Midas Workshop 2019

Discovery, accelerated

Overview

- The problem
- Experience from 3 experiments
- Potential for inclusion in core midas

The problem

- Experiments often have hundreds/thousands of settings in the ODB
- Want to be able to save/load these
    - Save time changing between different configurations
    - Ensure data is taken with a "known-good" setup
    - Limit the amount of bad data that is taken because an operator played with something and forgot to change it back

3 experiments

- I've implemented this twice, and am taking over responsibility for a third
  - DEAP
  - SuperCDMS
  - BNMR
- All experiments have different purposes/requirements for saving settings
- All 3 have completely different implementations
- Would be nice if midas could provide core tools

Experiment 1 - DEAP

- Purpose
  - Ensure data is only taken in a few known-good configurations (data type 456 has been used for WIMP-search data for the last few years)
  - Must specify data type 999 if you want to test/play
- Implementation
  - Few hundred ODB keys must be saved/loaded at once
  - Custom "start run" page won't let you start a run if ODB settings don't match what is expected (and will fix for you)
  - Saved settings stored as JSON in CouchDB (nosql database used for many other things in the experiment)

Experiment 2 - SuperCDMS

- Purpose
  - Help users to run with known-good settings
  - More "advisory" than DEAP - can still start a run if you've made changes
- Implementation
  - Several parts of ODB can be saved/loaded independently (e.g. "LED" settings X and "trigger" settings Y)
  - Settings stored in a MySQL database
  - Custom page allows saving/storing; calls JSON API implemented in python-based web server

Experiment 3 - BNMR

- Purpose
  - Help user load settings that have been used before
  - Just a helper - free to change settings as desired
- Implementation
  - Settings saved on disk in text files
  - Custom page calls perl scripts

Common requirements

- Define which bits of an ODB should be saved
    - E.g. everything beneath /A/B/C
- Storing and retrieving settings
    - Plus optional metadata (e.g. comment, who saved, when saved, ...)
- Apply saved settings to live ODB
- (Usually) a way to "diff" settings (either different sets of saved settings, or saved vs current)

Subtleties - ODB schema changes

- ODB structures change over time
    - Don't just simply `db_paste` the saved settings!
- Behaviour I implement
    - If old settings have a key that new ODB doesn't, ignore it
    - If old settings are missing a key that is in current ODB, leave it as is
    - Should inform user about these things so they can decide if we made the right decision!
- On DEAP, we automatically "fix" and update the saved settings with any new/removed ODB keys

Thoughts on implementing in core midas

- Shouldn't be too strongly opinionated
  - Experiments have different needs/use cases
- Should try to allow as many people to use it as possible
  - Don't rely on MySQL v5.8+, for example...
- Should try to keep separated from existing tools
  - E.g. don't add lots of new options to db_save()
- Ideally simplest version should work "out of the box", but provide APIs to allow experiments to build on

High-level proposal of what to provide

- C++ implementation
  - Class for defining which bits of ODB to save (and what metadata fields to store)
  - Functions to configure/save/search/get/delete
- mjsonrpc functions for all the above
- Basic webpage using mjsonrpc functions

Backend discussion

- How should we store all this?
  - In the history database
    - Requires implementing 4x search/save/... functions (FILE, MYSQL, ODBC, SQLITE)
  - In a separate database
    - Could limit to specific DB flavours, but more limiting for users
  - Just in files on disk
    - Maybe the easiest?

Summary

- Many experiments want to save/load ODB settings
- Purpose and usage varies greatly
- Propose a common set of low-level functions that experiments can build upon

- Comments, thoughts, suggestions?