Regression Testing Suite for Condition Data Storage

Simonas Joris Samaitis
University of Vilnius
09/13/2011

Outline

- Goal and features
- How it works
- Testing sequence
- Database
- Web interface
- Future improvements

Goal

- Automated testing of condition data formats
- Backward and forward compatibility
- Writing and reading data between different versions and architectures of CMSSW framework

Features

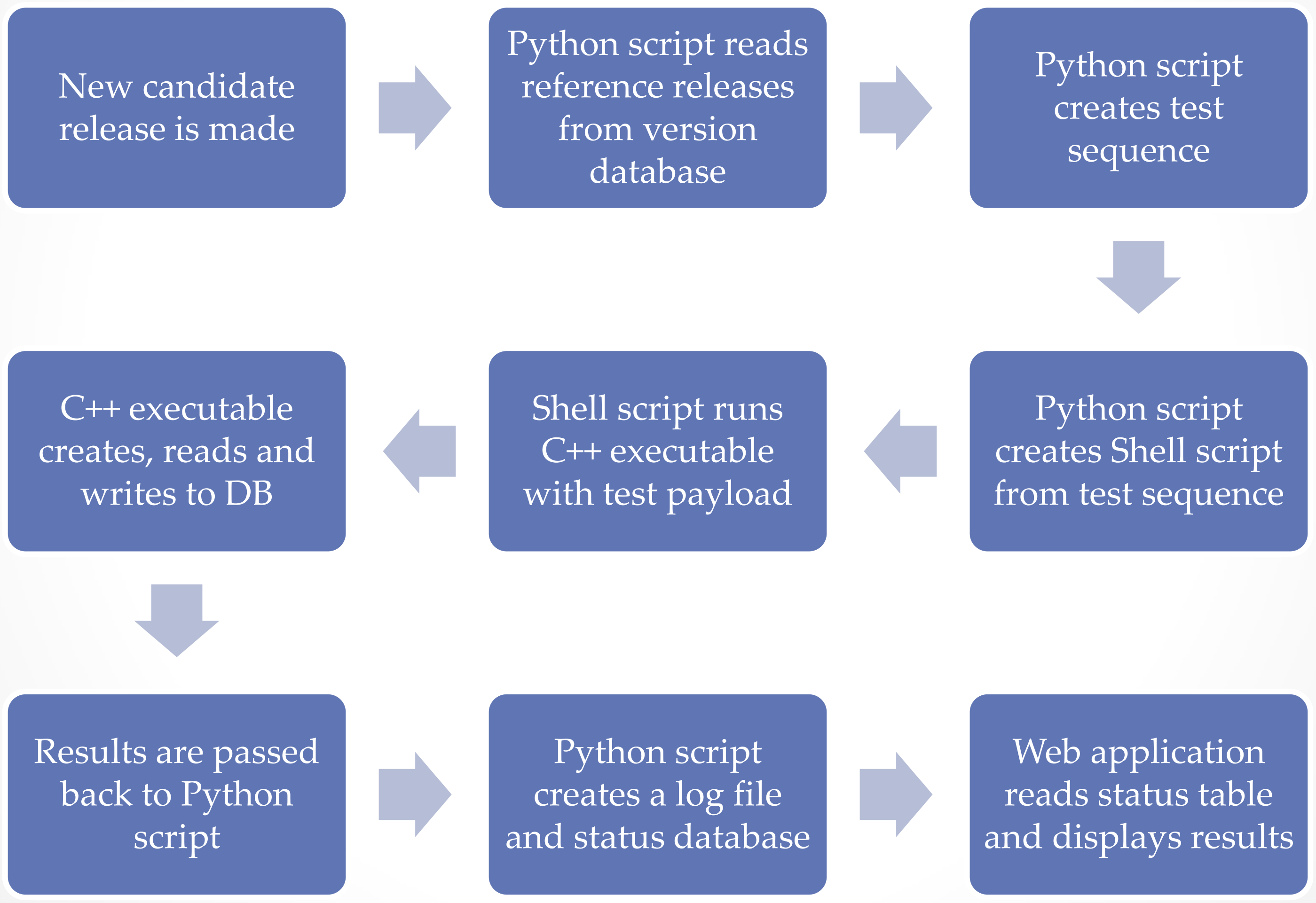
- Modular software design
- Monitoring over web application
- Test history logging
- Easy to extend to full condition data model

How it works

- As a new candidate release is made, it is tested against reference releases in database.
- Tests are done in pairs :

  (candidate release, reference release)
- Possibility to test only two recent builds

How it works part 2



New candidate release is made

Python script reads reference releases from version database

Python script creates test sequence

Python script creates Shell script from test sequence

Shell script runs C++ executable with test payload

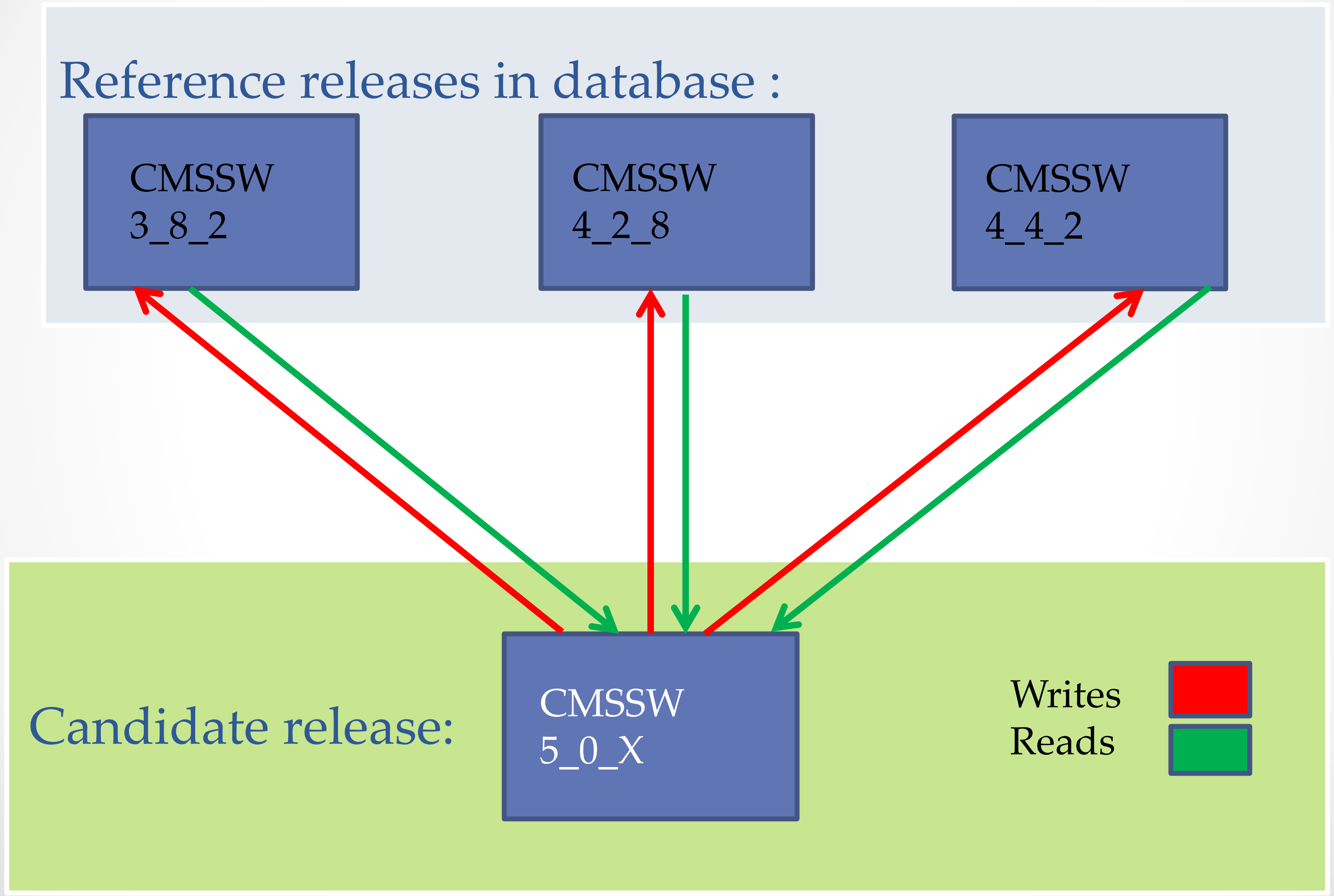
C++ executable creates, reads and writes to DB

Results are passed back to Python script

Python script creates a log file and status database

Web application reads status table and displays results

Test sequence



Test sequence in detail

- Set environment for reference release
- Drop all tables with reference release
- Create tables with reference release
- Write data with reference release
- Set environment for candidate release
- Read data with candidate release *
- Write data with candidate release *
- Read data with candidate release *
- Set environment for reference release
- Read data with reference release *

Segments marked with * represent key parts of the test, they are logged in test_status table.

Test payload structure

Simple types

- Integer
- Long
- Double
- Typedef
- String
- Enum

Data structures*

- Map
- List
- Pair
- Set
- Vector
- Structs

Class inheritances

Simple class tree in which one class inherits a few members from another

Example data structure : map of (string, Color) where Color is triplet of integers. More info in documentation

Database

Version_table

- Holds reference releases, each with respective architecture and path
- Accessed by pair (release, architecture)

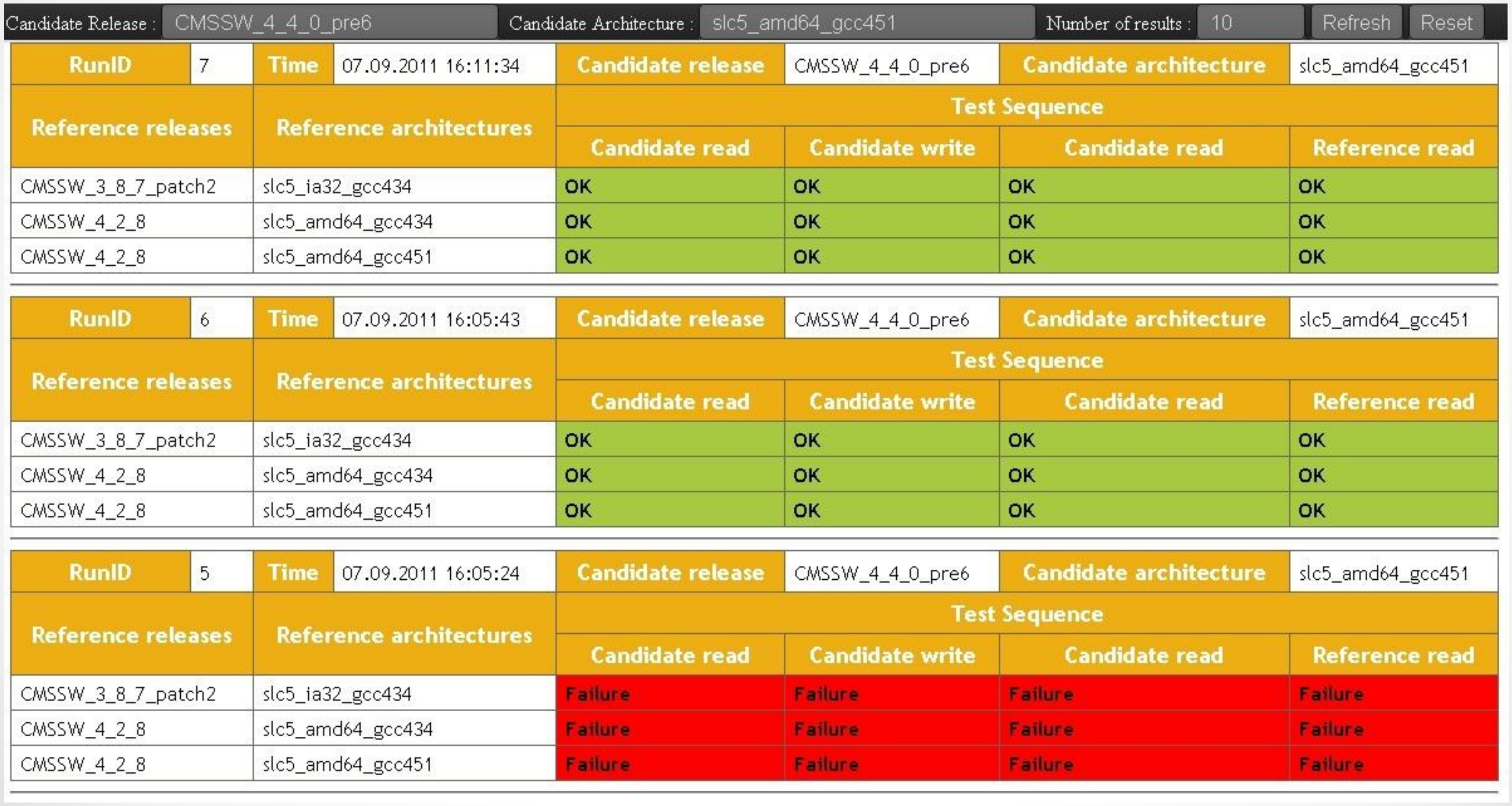
Test_status

- Holds the results of each test :
- Two pairs of (release, architecture) and 4 status codes for each test sequence

Web interface

- Runs on CherryPy
- Displays statuses of regression tests performed
- Search by test release, architecture

Web interface



Candidate Release : CMSSW_4_4_0_pre6 Candidate Architecture : slc5_amd64_gcc451 Number of results : 10 Refresh Reset

RunID	7	Time	07.09.2011 16:11:34	Candidate release	CMSSW_4_4_0_pre6	Candidate architecture	slc5_amd64_gcc451
Reference releases		Reference architectures		Test Sequence			
				Candidate read	Candidate write	Candidate read	Reference read
CMSSW_3_8_7_patch2		slc5_ia32_gcc434		OK	OK	OK	OK
CMSSW_4_2_8		slc5_amd64_gcc434		OK	OK	OK	OK
CMSSW_4_2_8		slc5_amd64_gcc451		OK	OK	OK	OK

RunID	6	Time	07.09.2011 16:05:43	Candidate release	CMSSW_4_4_0_pre6	Candidate architecture	slc5_amd64_gcc451
Reference releases		Reference architectures		Test Sequence			
				Candidate read	Candidate write	Candidate read	Reference read
CMSSW_3_8_7_patch2		slc5_ia32_gcc434		OK	OK	OK	OK
CMSSW_4_2_8		slc5_amd64_gcc434		OK	OK	OK	OK
CMSSW_4_2_8		slc5_amd64_gcc451		OK	OK	OK	OK

RunID	5	Time	07.09.2011 16:05:24	Candidate release	CMSSW_4_4_0_pre6	Candidate architecture	slc5_amd64_gcc451
Reference releases		Reference architectures		Test Sequence			
				Candidate read	Candidate write	Candidate read	Reference read
CMSSW_3_8_7_patch2		slc5_ia32_gcc434		Failure	Failure	Failure	Failure
CMSSW_4_2_8		slc5_amd64_gcc434		Failure	Failure	Failure	Failure
CMSSW_4_2_8		slc5_amd64_gcc451		Failure	Failure	Failure	Failure

To do list

- Deployment of automated testing of IB’s

Possible improvements :

- Extending to full condition data model
- Access to test logs from web interface