daq_slc
and
CVMFS

Nils Braun, **Oskar Hartbrich,** Mikihiko Nakao
B2 TRG/DAQ WS,
Yonsei University, 08/28/2019

daq_slc git repository

- daq_slc repo moved from B2G project to B2DAQ project
  - old repository URL still works (thanks to bitbucket auto redirect feature)
- Changed repo to strict PR-only policy (actually strictest repo in B2 for a short while)
  - Saw a **big** surge in contributions immediately
  - Code reviews very effective, often proved helpful to have more than one reviewer
  - Bamboo build system integration (automatic build test in PRs)
- Repo cleaned up very well
- git now widely adopted within DAQ group
  - Starting to use “advanced” git features (nsm2 as submodule)

Building daq_slc

- DAQ OS structure is heterogenous (SL5 on coppers, SL6 on ROPCs, SL7 on runcontrol machines)
  - Compilation of daq_slc was a pain, especially on coppers
- Cleaned up most Makefiles for sanity
  - Build instructions documented **in repository readme.md**, works on almost any machine now
  - Still not ideal, if someone wants to learn about make…
- Docker based build system: build daq_slc for all needed targets (SL5, SL6, SL7) on any (modern) machine
  - Guarantees **reproducible** builds in **validated environment**
  - Available and well documented: **B2DAQ/docker_build**
  - Pre-installed on central build server for everyone to use: build.daqnet.kek.jp, usage well documented **on Confluence**
  - I know of several active users of the system

CVMFS

- Now installed CVMFS infrastructure in daqnet
  - CVMFS: HEP standard for software distribution and version control (e.g. used on GRID)
  - Readonly (for users), **single source of truth**, integrated version control (easy to revert to previous version)
  - Mounted and used on all subdetector ROPCs, runcontrol machines
- Multiple mirrors in separate locations
  - No traffic/performance issues seen on mirrors
  - Mirror VMs had to be restarted by hand after KEK power cut
  - Still need to add mirror in B3
- No real issue/outage seen with CVMFS over past months
  - Extremely well documented **on confluence**
  - Currently distributing daq_slc, basf2 through CVMFS, open to user requests and suggestions. e.g. HSLB tools?

The daq_slc update cycle

- Current MO: **fresh compile** of current daq_slc master branch and publication on CVMFS, following **global restart** of all detector and runcontrol slowcontrol apps at the **beginning of each maintenance day**.
  - Fully automated **and documented** (expert) procedure
  - Can always do intermediate hotfix releases on request (~20minutes)
- Direct modifications of daq_slc folder not possible anymore, defined new local test procedure:
  - Build full binary packages from modified code on build server, copy to ROPC, selectively restart and test individual slowcontrol apps for testing
  - Step-by-step **instructions on confluence**
  - Successful local test demonstration is required to accept pull request into master branch
- Used daq_slc folder is monitored by Zabbix, warning when local test version is running on ROPC.
  - More sophisticated tools in development: now version string is hard-compiled into daq_slc libraries, can be read out through nsm
- daq_slc-slX binary repositories deprecated and practically disabled

Separation of daq_slc on CVMFS

- Proposed by Itoh-san, Yamada-san:
  Introduce different daq_slc CVMFS repos for copper/ROPC, HLT, runcontrol machines
  - Small change to single HLT config file currently means update of all subsystems
  - Keep same daq_slc source base
  - “Proper” release scheme with version numbers

My personal opinions: (very much open for discussions)

- The current standard operation is to compile daq_slc for all target architectures from current master, deploy to CVMFS, restart all daq_slc machines under DAQ control every maintenance day
  - The date + commit hash provided with each CVMFS version is effectively a git tag.
  - “proper versioning” should include a fixed set of successful validations. We have nothing of the sort in daq_slc,
  - We can deviate from this schedule and push a new version for quick fixes whenever needed. If a fix does not impact a given sub system, it is not necessary to restart it.
- But what about development tests? Make a temporary local version:
  - Need to change a config file? Local copy of cvmfs daq_slc version, modify config file, restart SLC from local directory.
  - Need to change the code? Clone repository on build.daqnet.kek.jp, compile using docker build environment (see slide 3), copy to your target machine, restart SLC from local directory.
  - For large scale tests: can publish test versions to CVMFS and not make them “default” (do not set “latest” symlink)

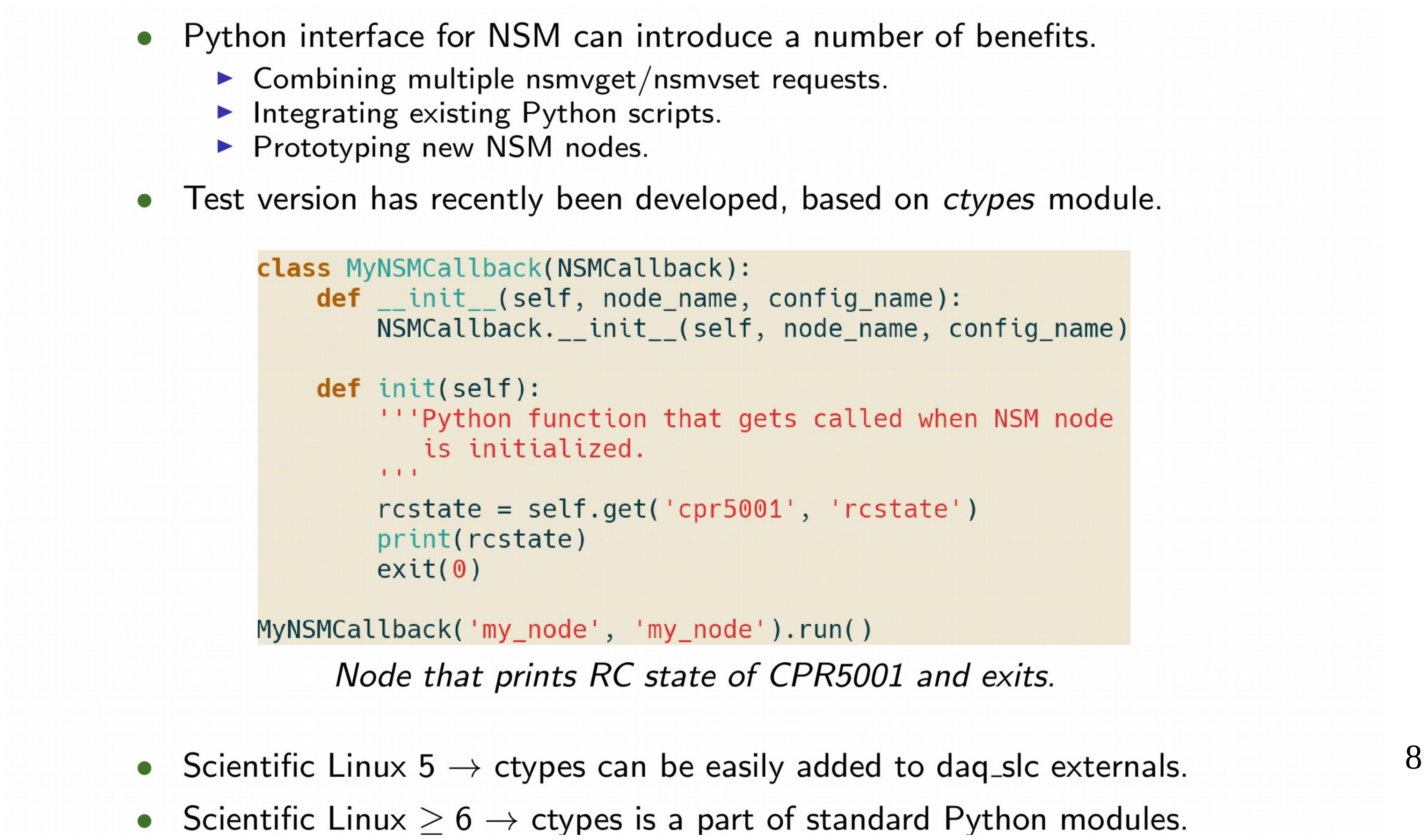
Towards the DAQ Upgrade

- As mentioned already: DAQ upgrade is a unique opportunity to improve on existing designs

  Some random thoughts:
- Will there be a canonic mapping cpr/HSLB → server/link? (I hope not!)
- cprcontrold
  - One thread per link? Current cprcontrold is link0-sequential in e.g. monitor requests, so everything would become slower in case of up to 48 links.
- rocontrold
  - one thread per link readout?
    (already answered: no. event building in DAQ upgrade module, one thread per connected module)
- DAQ configuration DB:
  - completely based on cpr/HSLB addressing, migration plans?
- Job management ("restart scripts") in DAQ upgrade?
  - See Nils' talk
- [not really SLC] Update of basf2 data structures and unpackers to support new raw format with extended buffers

Towards the DAQ Upgrade

- Maybe write some daq_slc parts in python?
- Mikhail Remnev has a “native” python binding:



- Python interface for NSM can introduce a number of benefits.
  - Combining multiple nsmvget/nsmvset requests.
  - Integrating existing Python scripts.
  - Prototyping new NSM nodes.
- Test version has recently been developed, based on *ctypes* module.

```python
class MyNSMCallback(NSMCallback):
    def __init__(self, node_name, config_name):
        NSMCallback.__init__(self, node_name, config_name)

    def init(self):
        '''Python function that gets called when NSM node
           is initialized.
        '''
        rcstate = self.get('cpr5001', 'rcstate')
        print(rcstate)
        exit(0)

MyNSMCallback('my_node', 'my_node').run()
```

Node that prints RC state of CPR5001 and exits.

- Scientific Linux 5 $\rightarrow$ ctypes can be easily added to daq_slc externals.
- Scientific Linux ≥ 6 $\rightarrow$ ctypes is a part of standard Python modules.

8

Summary and the Way Ahead

- The slowcontrol infrastructure has made a huge step forward since last B2GM, mostly thanks to technical expertise by Nils and continuous support by Nakao-san
- Infrastructure upgrade and repository cleanup has clearly contributed to the datataking stability
  - Several software improvements: Lock Guards, race conditions in DB provider

- Extend the monitoring capabilities of slowcontrol and nsm2 through Zabbix
- daqnet machines under DAQ control are integrated into CVMFS (etc.), but subdetector specific machines (conditions logging, power supplies) are not. How to proceed?
- Further future: What are the slowcontrol software infrastructure plans for the DAQ upgrade?