Using CMT in the LCG AA (nightly) builds

Andreas Pfeiffer
SPI

Outline

- Porting the AA projects
  - scram (V0) -> CMT
- Using CMT in LCG AA projects
  - Experiences with the nightly builds

Porting the LCG AA projects

- LCG AA review (Sep 06) recommended move from scram (V0) to CMT
  - Scram V0 no longer supported
  - Unclear future (maintenance) of scram V1
  - Guaranteed maintenance of CMT
    - Contract with Atlas
  - Basic functionality very similar
    - Different “syntax”

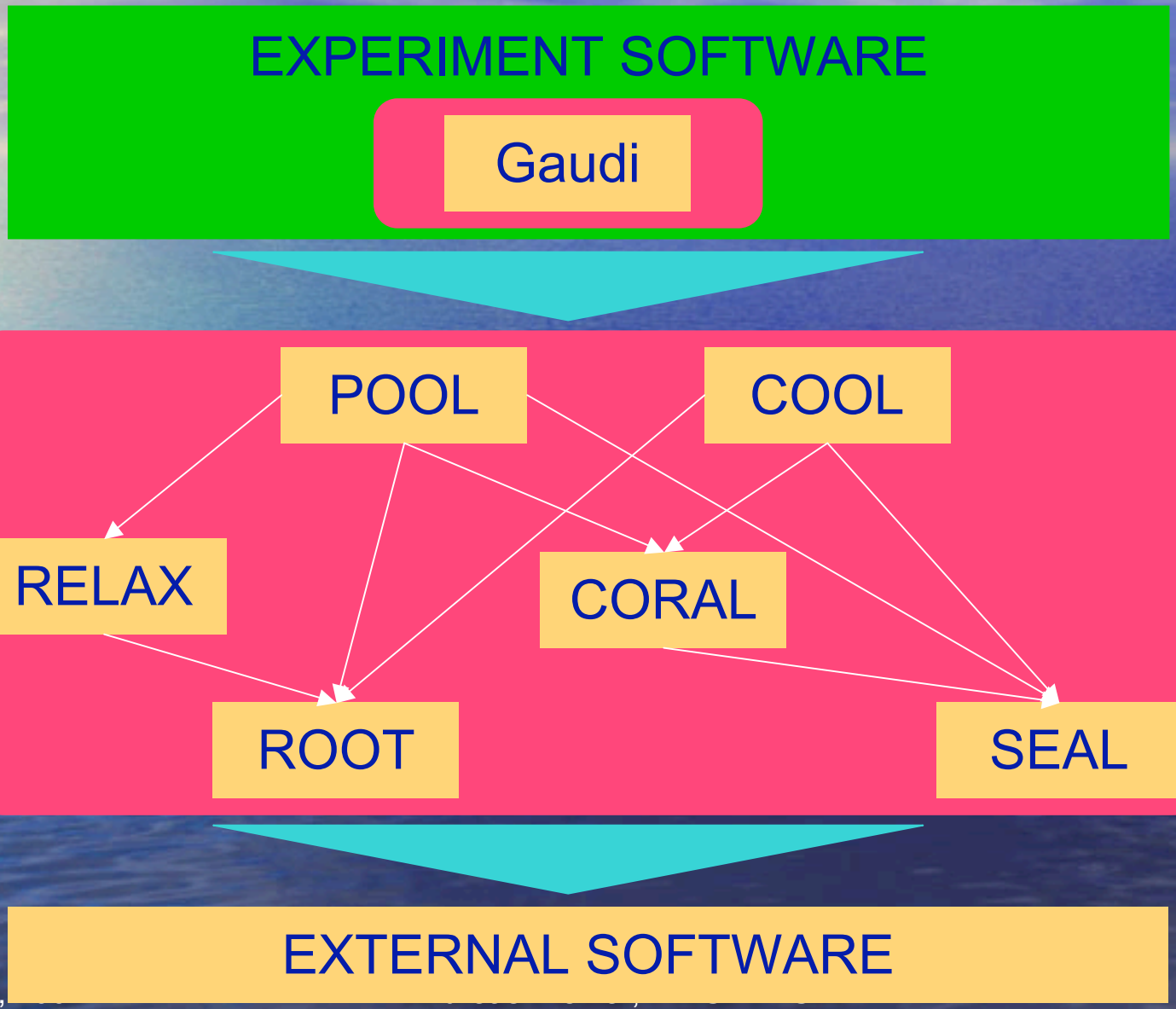
Making it work -- CMT porting (I)

- Started to port in the context of the nightly builds of LCG AA
  - Starting with SEAL (used CMT to build on windows already)
- Porting to RELAX, COOL, CORAL, POOL
  - In parallel with developers from projects

LCG AA nightly builds

- Main goals:
  - Provide prompt feedback of integrations and platform problems to LCG AA developers
  - Provide builds (binary) that the experiments can use directly to make their own tests
- *Validating the full stack of LCG AA s/w before releasing a configuration*

LCG Software Stack



Slots and builds

- "Slot" - defined as a set of CVS tags
  - Opened/closed on demand
  - Each slot defines the configuration selected
    - Via configuration file
- Cronjob builds each night
  - Start determined such that build is finished early morning (08:00)
  - Builds, run tests, install to AFS, analyze logfiles
  - Overwritten every week (Mon, Tue, …)

Nightly build system

- Implemented as a set of Python scripts
- Controlled by a configuration file
  - config.py
- Runs every night on all platforms
  - Via acron on linux, cron on Mac
    - AFS installation for Mac done via acron job on linux box (mounting the Mac's "/build/nightlies/")
  - Scheduled job on win (or WinAt) (not yet)
    - Work in progress
    - AFS installation similar to Mac

Output of the process

- Binaries build and installed in AFS
  - **/afs/cern.ch/sw/lcg/app/nightlies/<*slot*>/<*day*>/<*project*>/<*version*>/<*platform*>/**
  - LCGCMT is a project in there (installed per slot/day)
  - "stamp-file" to flag build is done (per platform)
- Tags in CVS for reproducible source builds (CMS)
  - "Running tags" and/or "fixed tags"
    - E.g. "LCGCMT_preview", "CORAL_1_7_0"
- Web page with status of all builds and tests
  - Static "summary page" available at: **http://lcgapp.cern.ch/spi/aaLibrarian/nightlies/index.html**

Making it work -- CMT porting (II)

- Cleaning up “windows specials” (in SEAL)
  - E.g. hardcoded use of “(x)copy”
- (Major) updates in LCGCMT
  - Introduction of “LCG_Configuration”
    - “One file to rule them all”: contains all versions for the given configuration, LCG_Interfaces/* refer to this one.
  - Using/updating/extending/creating new patterns
    - See later slide
- Few issues/problems found (and solved :-)
  - “unique naming” for tests
    - Using “postfix” to names for tests

Patterns in LCGCMT

- LCG_Policy, targeted at helping developers in projects
  - `pattern lcg_shared_library`
  - `pattern lcg_shared_generic_library`
  - `pattern lcg_module_library`
  - `pattern lcg_module_generic_library`
  - `pattern lcg_module_register`
  - `pattern lcg_application`
  - `pattern lcg_test_application`
  - `pattern lcg_test_library`
  - `pattern lcg_unit_test_library`
  - `pattern lcg_test_module`
  - `pattern lcg_unit_test_module`

Experiences Summary (I)

- Similar in complexity/functionality to scram
  - More “flexible” in some parts
    - “hacking the configuration” - adapting to new environments (e.g. nightlies, non-AFS builds)
  - Too “flexible” in others
    - Needs more strict control of environment
      - “CMTPATH”/”CMTPROJECTPATH” easy to “mess up”
    - No problem in nightlies
      - Maybe more of a problem for users/developers
      - Will get used to it :-)

Experiences Summary (II)

- Definitely needs more “reporting back” to user
  - Lost 2-3 days because I’ve tried to “apply” a pattern which didn’t exists (typo)
    - Error/warning would have saved a lot of time !!!
- Some “hardcoded” commands used in main CMT
  - “cp -a” -- “-a” not recognized on Mac
    - Now fixed, but a macro would be helpful

Summary

- CMT is becoming default build system in LCG AA projects
  - As of next configuration also POOL and COOL will be fully ported
- Successfully deployed in nightly builds for LCG AA projects (+Gaudi)
  - Some issues found, full list will be assembled and reported to developers
  - Main issue: need error/warnings for "missing" patterns, macros, …
- Use of nightly builds in Atlas, LHCb proved very efficient to find problems early !

Additional slides

Example “Use Cases” - “Slots”

- “Latest Greatest”
  - DEV of all
- “ROOT development”
  - ROOT DEV plus DEV of dependent packages, rest WORK
- “COOL development”
  - COOL DEV, rest WORK
- …

Selection of Use Cases depending on “hot” development areas, decisions taken in AF

Known issues

- Builds on non-AFS machines
  - no tokens in “cron”
  - Windows ???
    - Store in DFS ???
  - Mac OS X
    - “polling” data to AFS from linux
- Dealing with missing plug-ins (platform deps)
  - Makes analysis of log files more complex
    - Need to see where exactly the error is
  - No easy algorithm to decide if build is OK (stampfile)
    - Needs table of what should work on which platform

Present status

- Presently in set-up phase
  - Scripts for builds are basically working (Linux/Mac)
- Moving projects to build with CMT and QMtest
  - SEAL, RELAX, CORAL build now with CMT
  - Fixes also needed in LCGCMT
- Rudimentary logfile analysis at present
  - “webified” logs (warnings in blue, errors in red)

Near term planning

- Plan to have full stack by end next week
  - Builds and installs in AFS for experiments
  - Web pages with logs for developers
    - Static pages on build logs for a start
  - Running tests through QMtest
    - Needs adaption for CORAL, POOL, COOL
    - Complex testing environment !
  - Porting to Windows environment
- Analyze logs from running tests

Future enhancements

- Several (lots ?) of slots in parallel ?
- Build (and run tests) in parallel to speed up
  - More dedicated machines ? Grid ?
- "Dynamic" web pages
  - Colour code status of builds/tests
  - Needs handle on what is expected to build/run on each platform (plug-ins)