Modernizing the CMS software stack

Mircho Rodozov
On behalf of the CMS collaboration
Outline

➢ Introduction

➢ CMS Software stack
  ○ How big/complex is our stack

➢ Modernizing our stack
  ○ Python2-3 support
  ○ Code formatting

➢ CI/CD improvements & “exotic” experiments
  ○ Heterogeneous computing
  ○ Managing CI for 120+ github repositories
  ○ CI resources utilization
Introduction

➢ We are:
  - Core software @ CMS
  - Build releases, Integration builds (IBs), externals, test & deploy them
  - Continuous integration / delivery (CI/CD) system
  - Maintaining CI/CD infrastructure
CMS Software stack size

➢ Size of CMS Offline Software (CMSSW)
  ○ Over 4.5 M lines of code
    ■ 3.2 M C/C++ 1.4 M Python, 275k Fortran
  ○ Packages - 1300+
  ○ Multiple versions (10+ release flavors)

➢ Dependency on external packages
  ○ 500+, from source
  ○ custom build system (cmsBuild)

➢ Contributions
  ○ 200+ commits/week
    ■ 160+ github pull requests (PRs)
  ○ 800+ contributors

➢ 200+ GitHub repositories for CMSSW and externals
Software Distribution rate

➢ Release builds
  ○ Every two weeks
  ○ via github issues

➢ Integration Builds
  ○ ~450/month

➢ Complexity

➢ Extensive validation of IBs
  ○ 4000+ tests

Modernizing the CMS software stack
CI setup

DQM: Reduce TH1 Usage #28342

submit PR

please test

+1
Tested at: 3ffdf28

DQM: Reduce TH1 Usage #28342

submit PR

please test

+1
Tested at: 3ffdf28

Modernizing the CMS software stack
Code format

➢ Clang format and clang tidy
  ○ Formatted 3.2 M lines
  ○ Collision with open PRs avoided
  ○ Style used: Google
  ○ Rules enforced via PR testing

➢ Automated transition

Running code-format for pdmv #27357

Clang format

Automated transition

Running code-format for CMS5W category pdmv.
See the build logs here https://cmsdiit.cern.ch/jenkins/job/GitHub-refactor-cms5w-module/4390/console

cms-bot has successfully run the following:
  • scrum build code-check-all
  • scrum build code-format-all
  • scrum build

Clang tidy
Modernizing python

➢ 1.2 M lines converted to be py3 compatible

➢ Migration completed
  ○ Made it compatible while running py2
  ○ Works with both

➢ Mostly a manual job
  ○ Limited number of cases
  ○ Some conversion tools exist (futurize, autopep)
Python packaging upgrade

➢ Integrated 200+ python externals
  ○ Build from sources using pip

➢ Simplified format
  ○ one line update
  ○ py2 and py3 recipes unified

➢ Customization for the new packages
  ○ a package can have patch
  ○ build only for py2 or py3 if required
  ○ conditionally build
    ■ python version
    ■ architecture

Modernizing the CMS software stack
Validating externals

➢ Integration tests for externals
  ○ Does it integrates (build + link against the IB)
  ○ Does it run

➢ Physics validation
  ○ Comparison with reference

Backport OpenBLAS fixes [10_6_X] ✓ comparison-available externals-pending orp-pending
pending-signatures tests-approved
#5291 opened 9 days ago by kpedro88

Backport OpenBLAS fixes [10_2_X] ● comparison-available externals-pending orp-pending
pending-signatures tests-rejected
#5289 opened 9 days ago by kpedro88

Comparison is ready
https://cmscdt.cern.ch/IDTjenkins-artifacts/pull-request-integration/PR-de759a330223/summary.html

Comparison Summary:
- No significant changes to the logs found
- Reco comparison results: 0 differences found in the comparisons
  • DQM/HistoTests: Total files compared: 33
  • DQM/HistoTests: Total histograms compared: 3212324
  • DQM/HistoTests: Total failures: 1
  • DQM/HistoTests: Total nulls: 0
  • DQM/HistoTests: Total successes: 3211989
  • DQM/HistoTests: Total skipped: 334
  • DQM/HistoTests: Total missing objects: 0
  • DQM/HistoSizes: Histogram memory added: 0.0 KiB (32 files compared)
  • Checked 137 log files, 14 edm output root files, 33 DQM output files
Testing multiple PRs

➢ Testing 200+ different repos (CMSSW, externals, data)
➢ Testing multiple PRs from different repos at once
➢ Convenient for testing pull requests that need external update

Modernizing the CMS software stack
CMSSW on heterogeneous resources

➢ ARM
  o Cavium ThunderX 96 cores
  o Cavium ThunderX2 128 cores

➢ PowerPC
  o Power 8
  o 1 Wistron 256 core

➢ GPUs
  o Shared Nvidia Tesla K20X
  o HTCondor GPUs resources
    ■ as Jenkins nodes
    ■ released on job completion
    ■ Thanks to IT-CM for adding a short queue to access these nodes for us!

Modernizing the CMS software stack
CI resources utilization

➢ We run our own scheduler for CI validations
➢ Heuristics based, 20 to 25% faster
➢ https://tinyurl.com/yywh2py7
Conclusions

➢ CMS improves its code base continuously
➢ Keep improving our CI/CD system
➢ Embraced modern code policies
➢ Migrated python
➢ Successfully running CI and tests on non Intel architectures
➢ Put some effort for better utilization of the CI resources