Software Release Building and Validating in CMS

Andreas Pfeiffer, CERN PH

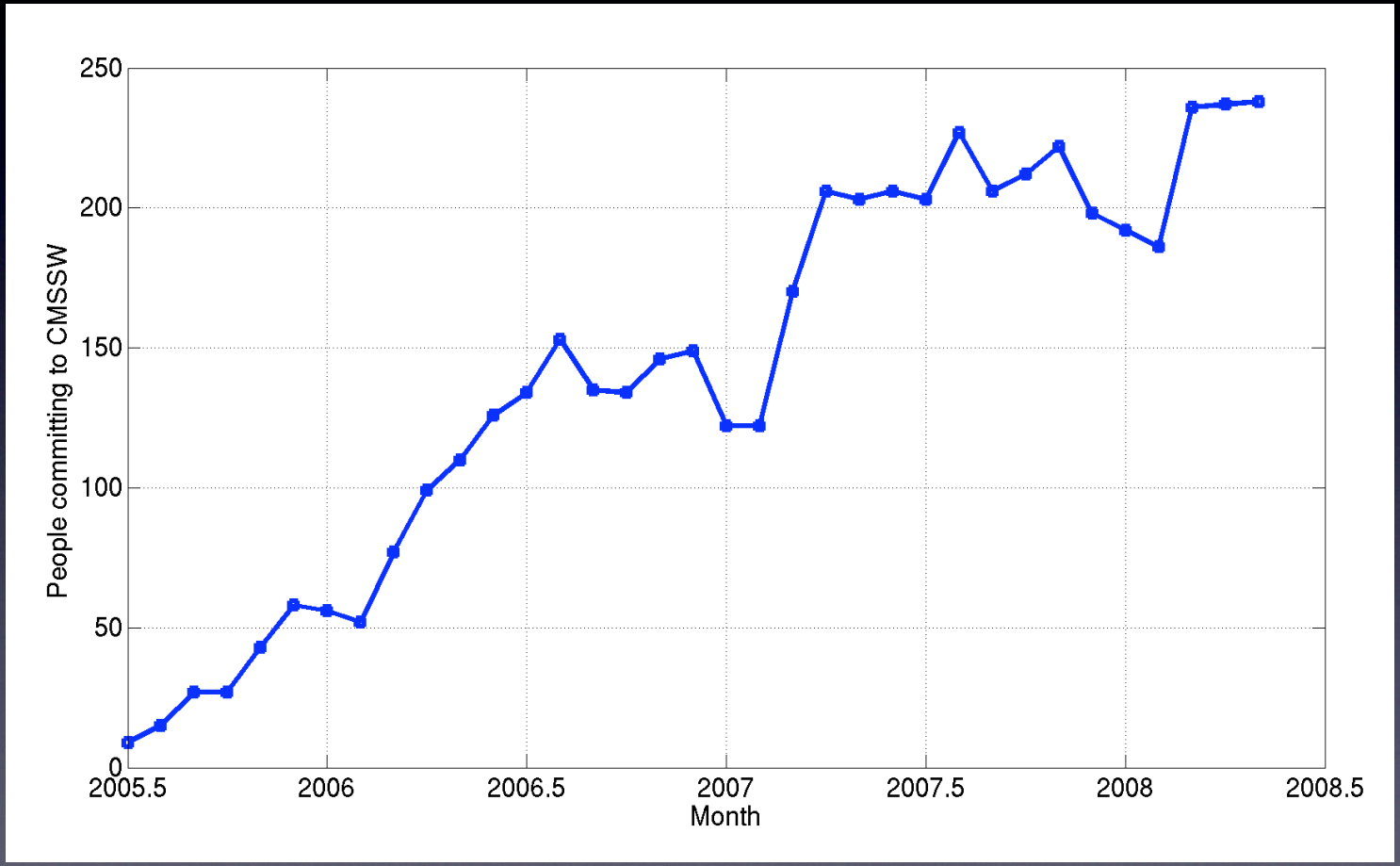
Outline

- CMS software
- Development model and tools
- Release process
- Validation

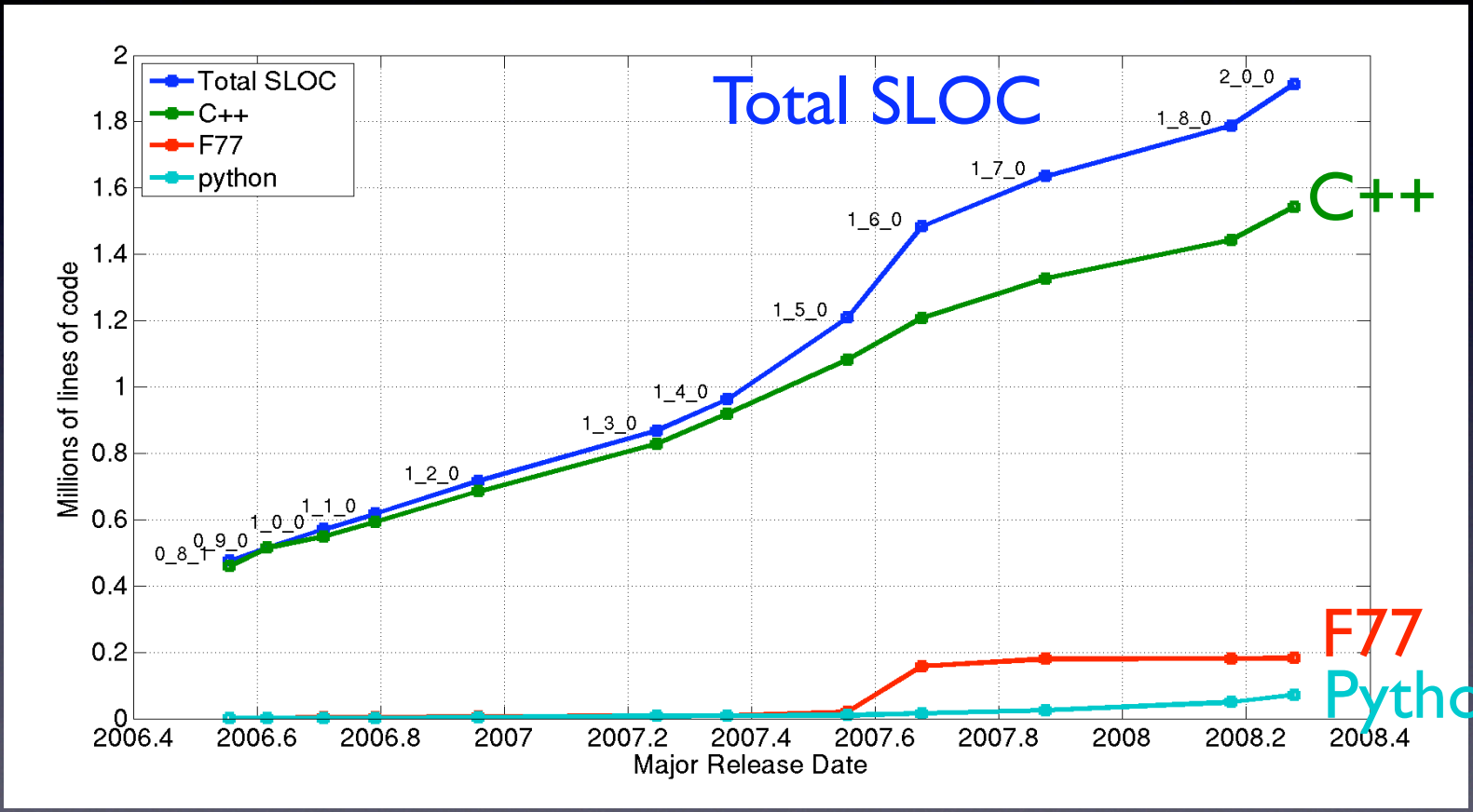
CMS software

- ca. 1100 individual Packages organised in 100 SubSystems
- ca. 2 MLOC
- 250 active developers
- ca. 100 external packages
- ca. 1.5 GB of "data" packages
  - mainly for FastSimulation

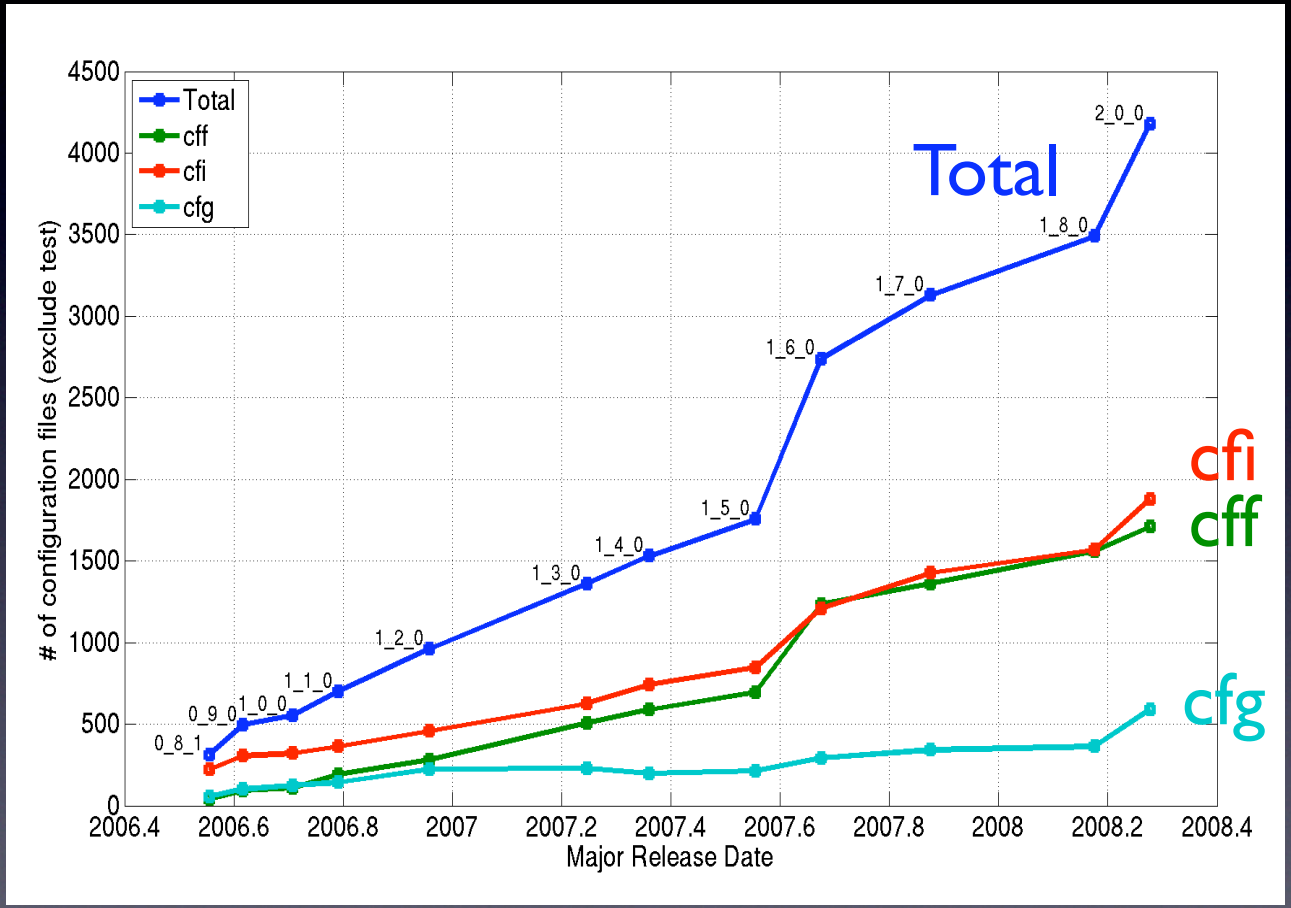
Number of Developers Committing to CMSSW



Source Lines of Code



Number of Configuration Files



External packages

- ca. 100 externals (incl. LCG AA projects)
- Building and installing new versions as needed
  - using patches as appropriate
- Complex dependencies handled "automatically" via spec files

Development model

- Always the full set of packages has to build
  - partial relases done later (FWLite, online)
- Integration Builds
- Development Releases
  - "Open" and "Closed" phases
- Production Releases

Development Tools

- Building using SCRAM configuration management and build tool
  - version 2.0.5
    - major performance boost
      - O(10) in memory usage, start-up time and disk usage
- *addpkg* and *checkdeps* scripts to check out packages and their dependencies

Release Tools (I)

- Build and install based on *rpm* and *apt*
  - Well proven tools (linux distributions)
  - Excellent dependency checking/verification
  - Customised to use private DB and non-root install
  - We hit the limits with our use-case
    - Several releases installed hit limit in rpmDB (too many files)
    - Need to limit the number of installed releases

Release Tools (II)

- *cmsBuild* (python script) to check consistency and build release (and all external packages) in coherent way
  - using spec files (customised)
  - patches allow flexible adaption of external packages
  - building in parallel (and with make -j)
    - 4 h building externals (incl. gcc, python, g4, root)
    - 2.5 h building CMSSW
- *rpmbuild*: bottleneck (1.5 h single thread after build)

Release scheduling

- Development ("pre") releases about every two weeks (for X.Y.0, later only if needed)
  - "open" and "closed" phases
- Production releases
  - planning wiki page
- Discussing "Analysis Releases" for the future

Major Software Releases

- CMSSW_1_8_0 released early March
  - Used in Global Runs
  - 10 development releases
  - 5 production releases
- CMSSW_2_0_0 released early May
  - Used in CSA08 and Cosmic Runs
  - 9 development releases
  - 12 production releases

Major Software Releases

- CMSSW_2_1_0 release imminent
  - used for data taking
  - 11 development releases
- Future 2_X_Y releases will be managed under "closed conditions" to guarantee *correct content* and *stability*
  - agreed list of changes
  - new tags checked and approved by convenors

Release process

- Integration builds (IB)
- Development releases (-preN)
- Production releases

Integration Builds (I)

- Former "Nightly build"
  - Change of system in summer last year
- Two IB per release cycle per platform
  - Only one official platform (slc4_ia32_gcc345)
  - Several cycles in parallel
    - CMSSW_2_0_X, CMSSW_2_1_X, CMSSW_3_0_X

Integration Builds (II)

- Release manager (one per cycle) follows up problems in IB with developers
  - via HyperNews (daily updates)
  - checking status of build and tests in detail
- Web portal for results of build and tests
  - Used to ease communication (sending URLs)
  - Needs improvements in usability

Testing in Integration Builds

- Two types of tests in IBs
  - Unit tests, executables
    - presently run in IBs
    - based on *configuration files*
      - *cmsRun <cfg>.py*
      - not yet in IBs, run manually by developers

Development Releases

- Open phase: time-driven
  - "take the tags from an IB and make it work"
  - validation using reduced set of tests
- Closed phase: feature driven
  - find and fix bugs
  - allow (some) new features (controlled)
  - validated using extended set of tests

Release Validation

- Create physics events for different samples
  - Single particles, MinBias, specific physics channels
  - variations in geometry, field, beamspot
  - customisable
- Two "classes" of RelVal samples
  - "standard" 10k events - 24h turn around time
  - "high-stat" 25k events - 1 week turn around time

Release Validation stages

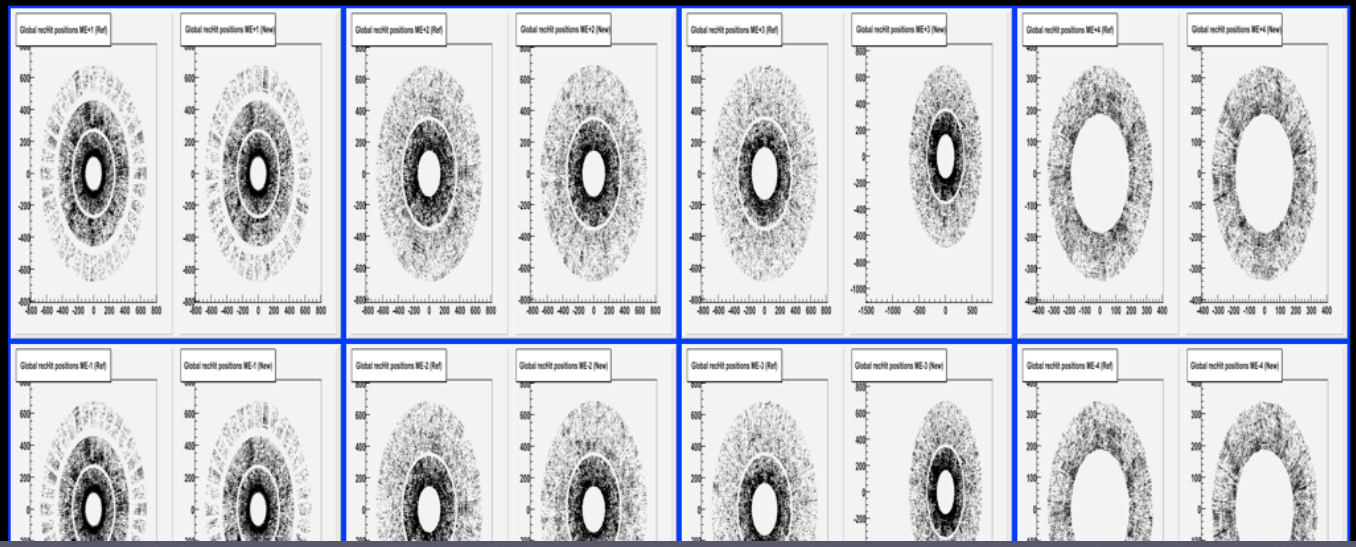
- Integration Builds
  - run subset of RelVals with 10 events (23 samples)
- Open Development Releases
  - run standard and high-stat samples with 10 events
- Closed Development Releases
  - run full standard and high-stat samples
  - feedback from developers community
  - performance studies

Validation pages

CMSSW Validation plots for 100 GeV single muon Relval Sample (CMSSW_2_1_0_pre9)

Release: CMSSW_2_1_0_pre9
Reference: CMSSW_2_1_0_pre6
Run #: 100 GeV single muon Relval Sample (CMSSW_2_1_0_pre9)
Reference: 100 GeV single muon Relval Sample (CMSSW_2_1_0_pre6)
CSCValidation was run on 23-July-2008

RecHit Global Positions

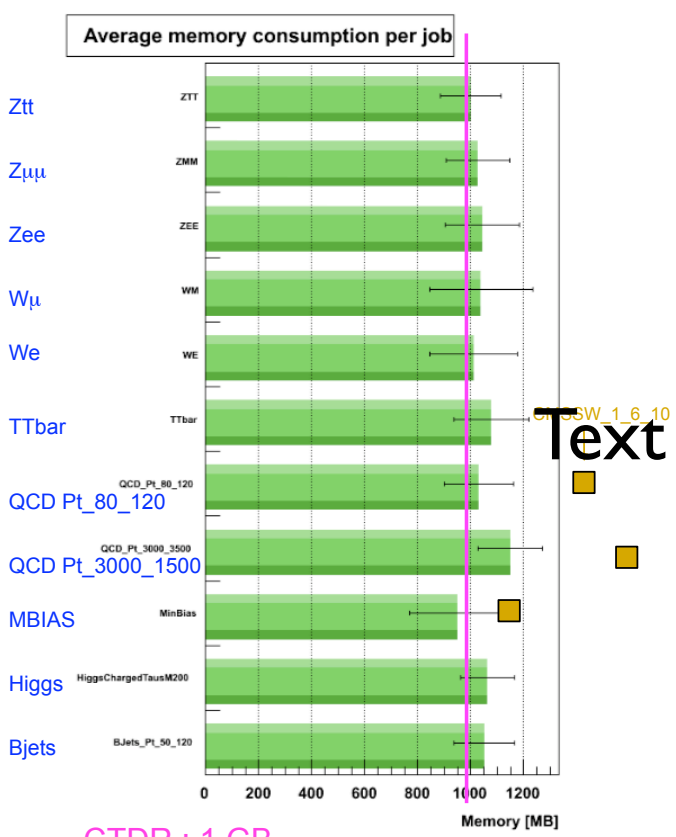


Performance Studies

- Performance benchmarking part of the validation process
  - Using RelVal samples
  - Using "standard candles" defined by Simulation and Reconstruction teams
    - HiggsZZ4l, MinBias, QCD80-120, TTbar, Single particle (e, pi, mu)
- Follow up with developers as soon as changes are detected

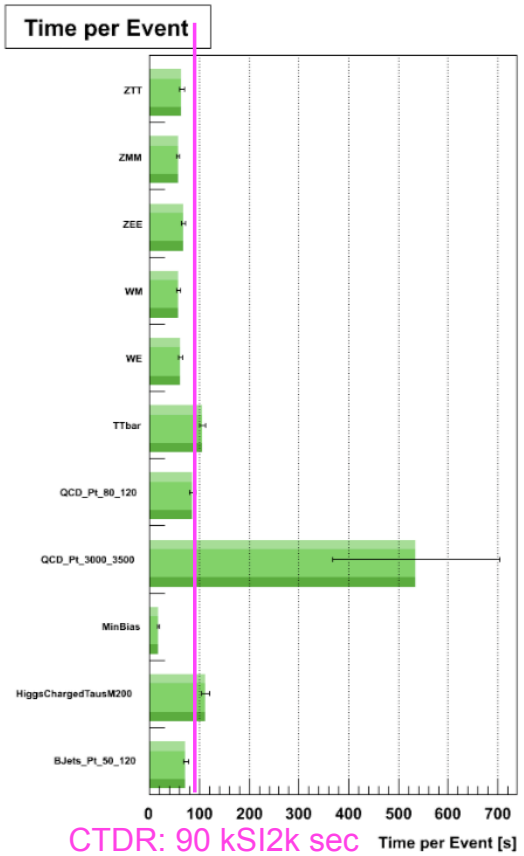
Performance (I)

2_1_0_pre5 : GEN+SIM+DIGI+L1+DIGI2RAW+RAW2HLT



CTDR : 1 GB

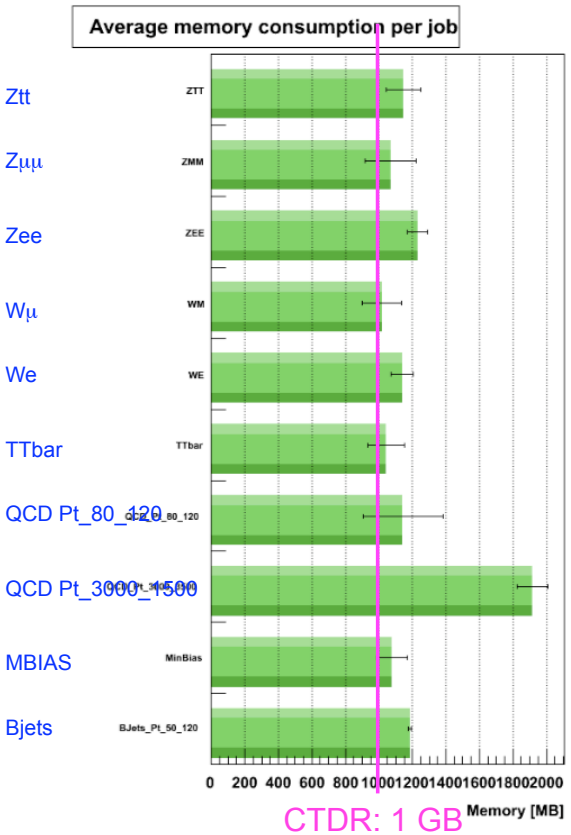
Text



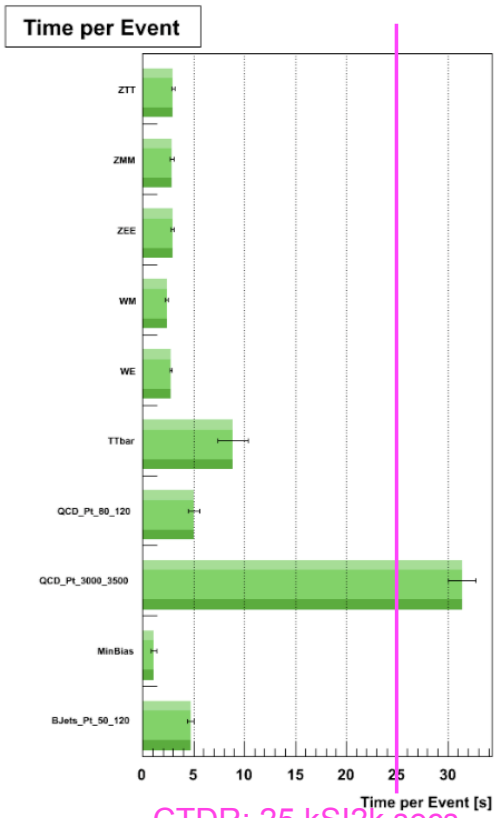
CTDR: 90 kSI2k sec

Performance (II)

2_1_0_pre5: RAW2DIGI + RECO



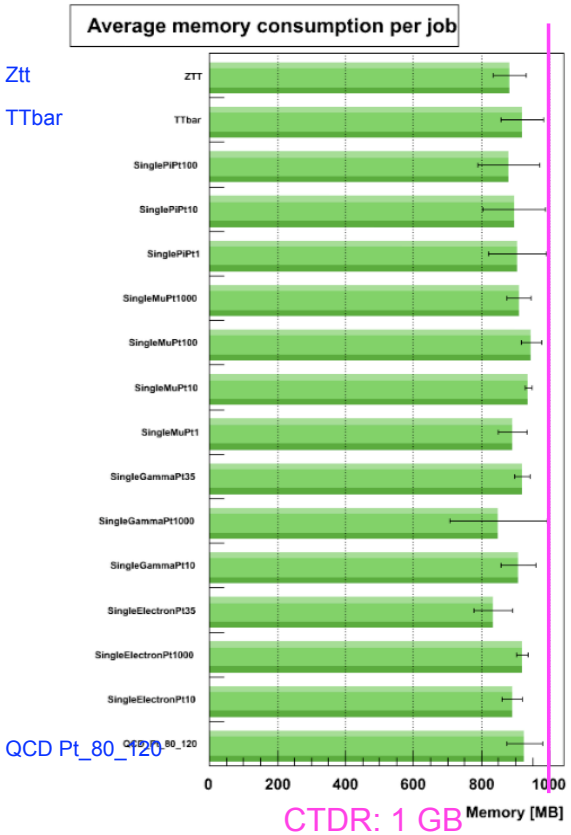
CTDR: 1 GB



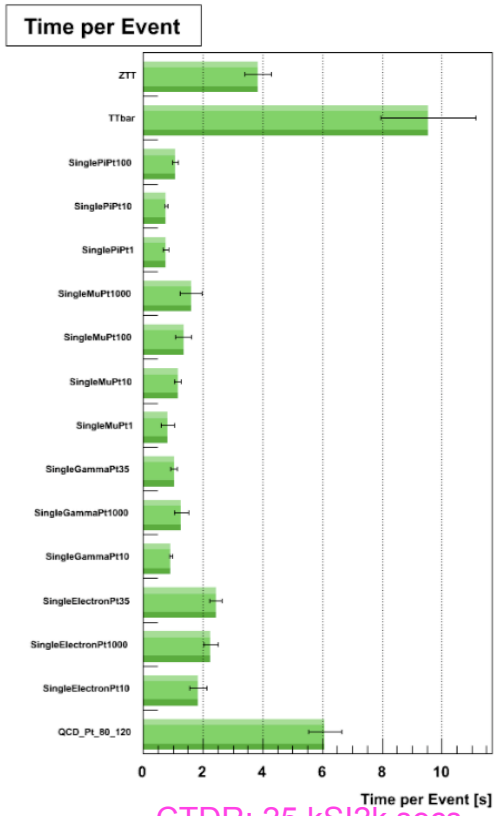
CTDR: 25 kSI2k secs

Performance (III)

2_1_0_pre6: RAW2DIGI + RECO



CTDR: 1 GB



CTDR: 25 kSI2k secs

Performance: Work in Progress

- Framework team continuing to monitor and fix code that contributes to poor performance
- Memory
  - ROOT IO buffers: options include switch to non-split mode for RAW/RECO data, reducing basket size, 'drop on input' etc.
  - Library size (240 MB): remove unnecessary dependencies e.g. ORACLE
  - Python: 15-30MB not de-allocated after configuration step
    - Move to python 2.5 will reduce this

Performance: Work in Progress (II)

- CPU performance
  - Memory (de)allocations: 20% cpu used in memory ops (new/delete)
- Data size
  - Size of event metadata increased by factor x4 since 2_0_0 to ~40kB/event (depending on samples)
  - Back to "normal" in 2.1.0-pre9
- Studying to use a few "big libraries" instead of one (few) per package

Summary

- CMS software development is a complex task
- Release process in several stages
  - controlled stages
  - validation at each stage at different scales
  - developers feedback on the RelVal samples