

CMS software

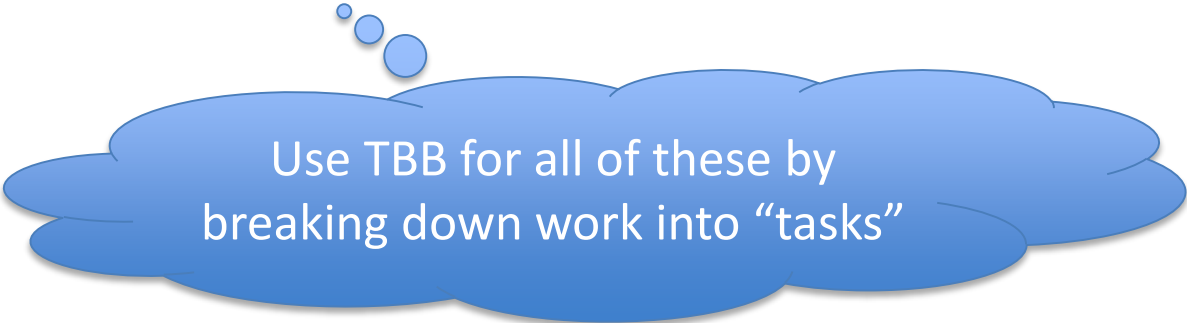
David Lange
June 6, 2016

Outline

- Threading in CMSSW
- Git / GitHub / Integration infrastructure
- Simulation and Geant4
- Conditions

Threaded CMSSW framework design

- Run multiple data taking transitions in parallel
- Run multiple modules concurrently within one event,
 - Change to user code: Needed more information about module dependencies: Declare what data products a module will consume in addition to what it will produce
- Run multiple tasks within a single module concurrently



Use TBB for all of these by breaking down work into “tasks”

Framework implementation: Thread safety requirements

Data Products

- Information passed from module to module
- Only const access to data products is provided
- **const member functions must be thread safe** (Matches C++11 thread-safety guarantee for containers)

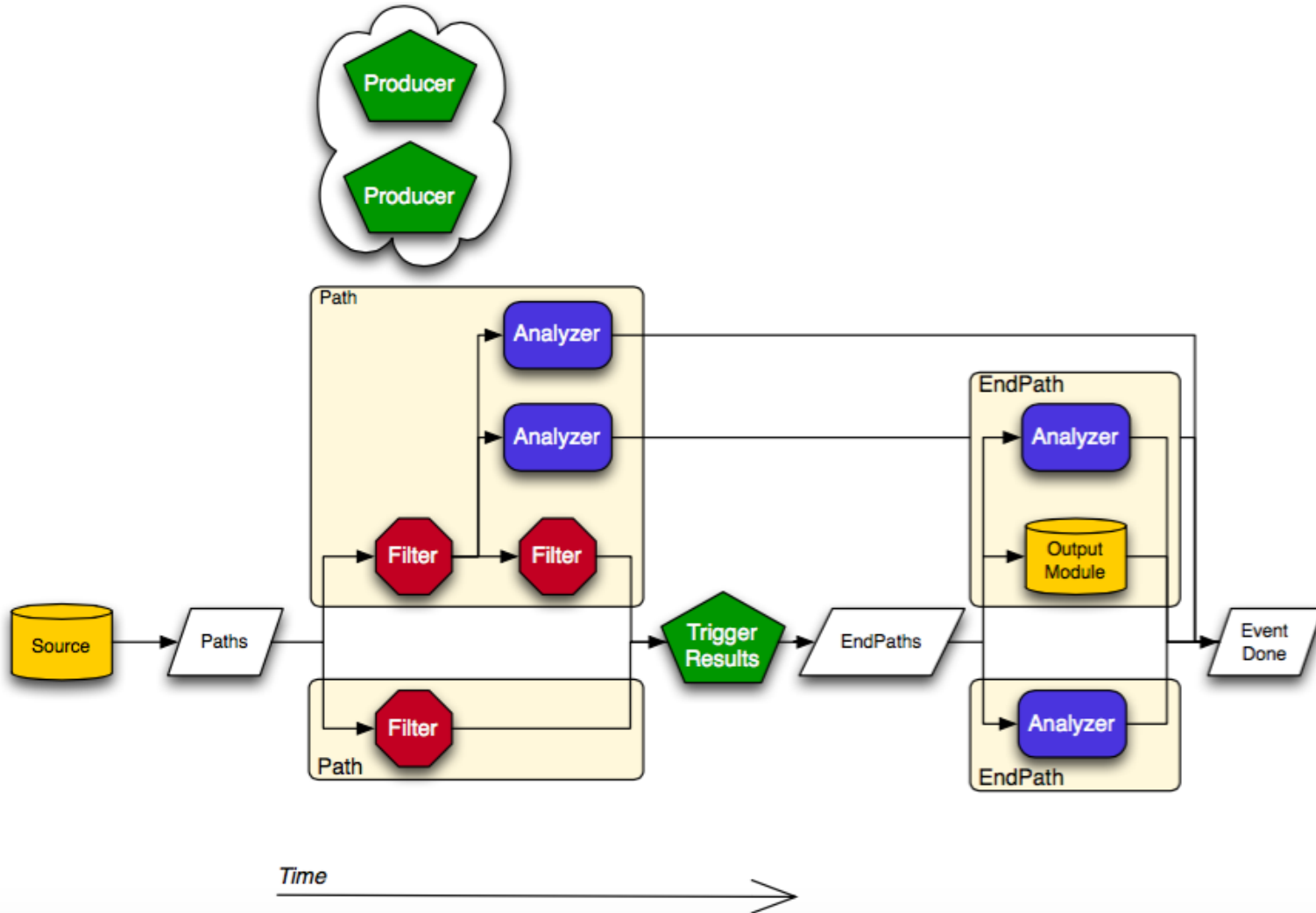
EventSetup modules (primarily conditions information: IOV driven)

- EventSetup using one mutex
- If an EventSetup modules needs to run, the lock is taken. However, accessing cached data does not require a lock

Producer, Analyzer, Filter modules

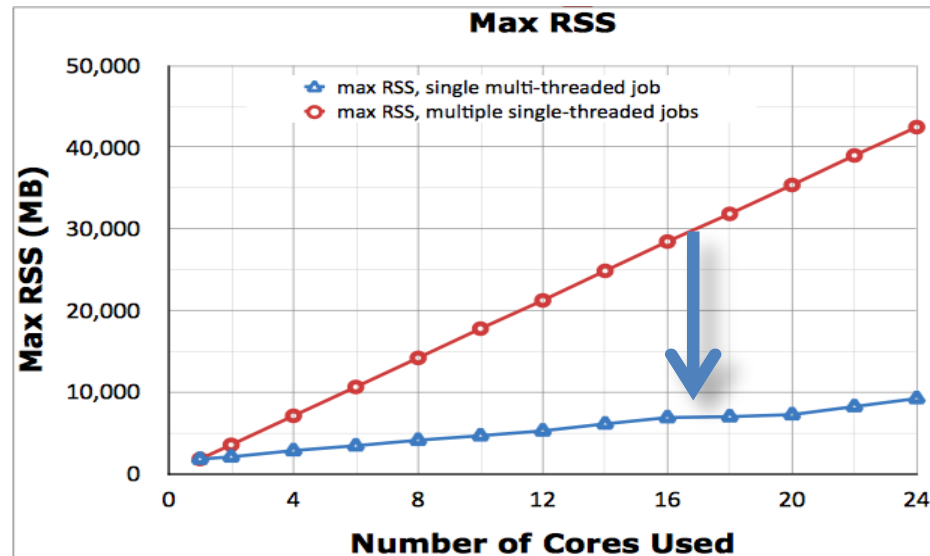
- Majority of user written code
- Module base class options define thread safety requirements
 1. **Legacy**
 2. **Stream**: One copy of module per stream (thread)
 3. **Global**: Reentrant, sees all events
 4. **One**: Shared by all streams (not thread safe)

Threaded CMSSW Framework concept



Multithreaded status

Example RSS savings from threading in CMS (reconstruction)



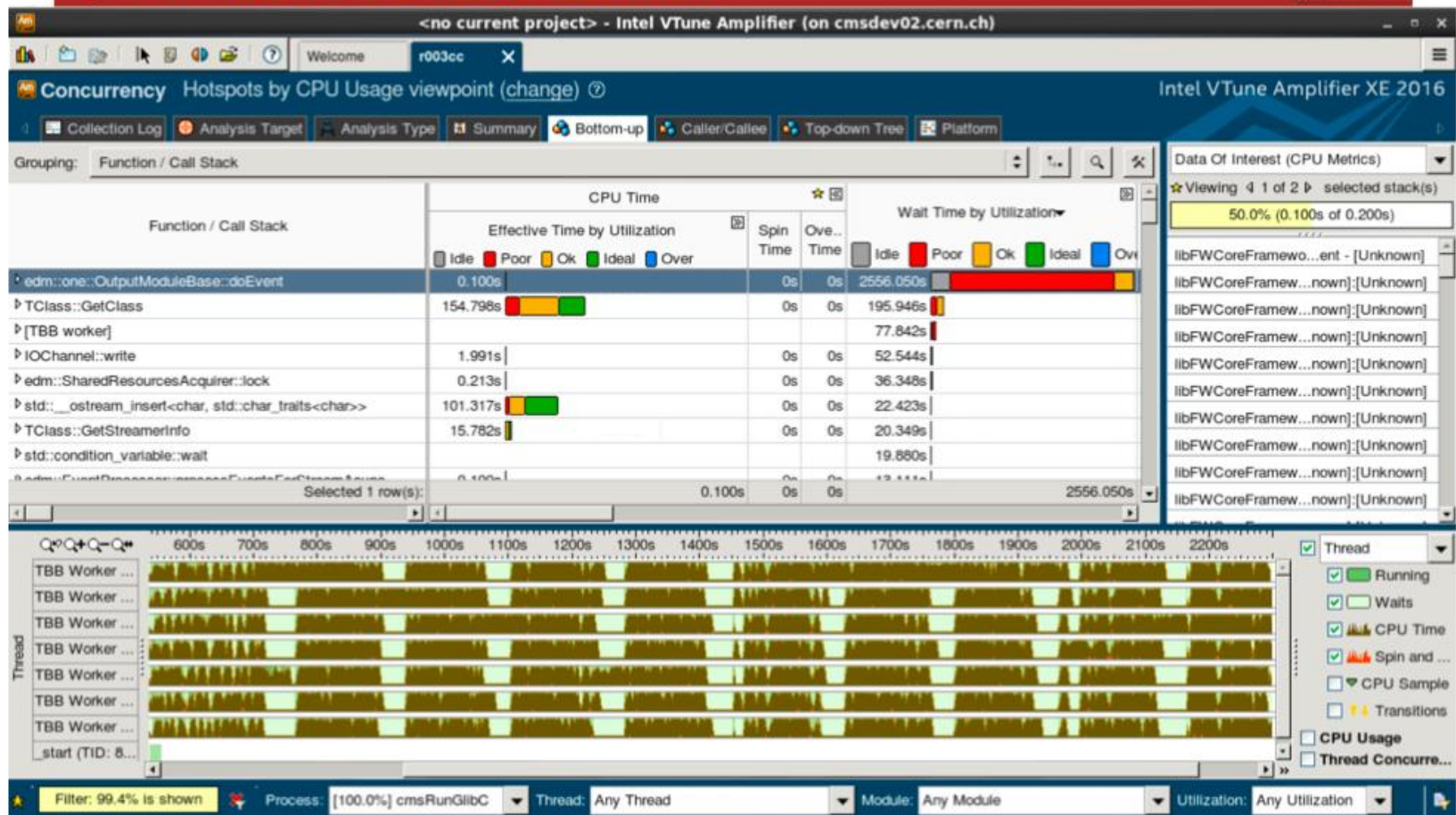
- Status of our main workflows in production
 - 2015: Tier-0, HLT, data reconstruction run multi-threaded
 - 2016: All major workflows are able to run efficiently in multithreaded mode. Still working through deployment details
- Framework development goals for 2017
 - Parallel running of modules within an event
 - Parallel running of events in multiple lumi sections

Some lessons learned

- Approach based on different flavors of algorithms (legacy, stream, one...) has eased the transition to production
 - Even simple interface changes prove to take a long time to complete (in CMS at least).
 - Debugging still largely a core SW group task: Fortunately we have not experienced major or extremely rare problems
 - Optimization also largely a core SW group task, but CMS tools for identifying bottlenecks are improving
 - Identify modules responsible for stalls
 - Helgrind
 - Static analysis

Threading optimization: We use VTune very successfully

Final Results



GIT TRANSITION AND WORKFLOWS

CVS → GIT transition for CMSSW

- Transition motivated by the end of CVS repository hosting support at CERN
[Transition completed summer of 2013]
- After an evaluation of different options (SVN, CERN hosted Git), we migrated the CMSSW code repository from CVS to **GitHub**

CVS → Git transition

- Repository structure: We stayed with one repository for all of CMSSW
 - We did not see a way to split the repository in a way that would not allow most requests to be against just one repository
 - Given 1100+ packages, we defined a mapping between code chunk (“packages”) and software conveners responsible
- Repository structure
 - One branch per release cycle plus branches as needed for operational bug fix release builds
 - Handful of people that can integrate code
- We moved beyond nearly all of the CVS specific utilities we had developed during Run 1 (not initially, but over time)
 - Using the gitHub API to drive request, testing and integration procedure

CVS → Git transition

- Development history:
 - We kept old official release tags from CVS but not the private tags that we allowed in CVS packages
 - Full file history is preserved (even if not trivial to access)
- Data files: We moved all sizeable data files into separate repositories to keep the CMSSW repository size manageable.
- Caching our repository locally helps considerably

Code request lifecycle (example)

- User makes a pull request to the cmsw github repository

The screenshot shows a GitHub pull request for the repository 'cms-sw:CMSSW_8_0_X'. The pull request is titled 'L1TRawToDigi Fixes #13607' and is created by 'Martin-Grunewald'. It shows 8 commits and 5 files changed, with a net change of +30 lines and -22 lines. The pull request is currently in a 'pending' state, as indicated by the 'orp-pending' label. The pull request description states: 'L1TRawToDigi Fixes Includes a fixed version of @mulhearn 's #13549 and thus replaces that PR; and a fix by @blwiner on the number of algos being unpacked.' The pull request is assigned to 'cmsbuild' and is part of the 'Next CMSSW_8_0_X' milestone. The pull request is also labeled with 'comparison-pending', 'pending-signatures', 'tests-pending', and 'orp-pending'. The pull request is currently in a 'pending' state, as indicated by the 'orp-pending' label.

L1TRawToDigi Fixes #13607 Edit

Open Martin-Grunewald wants to merge 8 commits into cms-sw:CMSSW_8_0_X from Martin-Grunewald:L1TRawToDigiFixes

Conversation 23 Commits 8 Files changed 5 +30 -22

Martin-Grunewald commented a day ago cms-sw member

L1TRawToDigi Fixes
Includes a fixed version of @mulhearn 's #13549 and thus replaces that PR; and a fix by @blwiner on the number of algos being unpacked.

mulhearn and others added some commits 5 days ago

- Change empty payloads from Error to Warning, and stop after 5 66eb1d1
- Merged refs/pull/13549/head from repository cms-sw 4c8d048
- fix max algo comparison 6adb4aa
- L1TRawToDigi unpacker fixes a8f454b

cmsbuild added this to the **Next CMSSW_8_0_X** milestone a day ago

cmsbuild added **l1-pending** **pending-signatures** **tests-pending** **orp-pending** **comparison-pending** labels a day ago

Labels

- comparison-available
- fully-signed
- l1-approved
- orp-pending
- tests-approved

Milestone

Next CMSSW_8_0_X

Assignee

No one—assign yourself

Notifications

Unsubscribe

You're receiving notifications

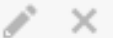
Code request lifecycle (example)

- Standard tests are requested (by “known” users)




Martin-Grunewald commented a day ago

cms-sw member



please test

- Comparisons are returned for evaluation by category managers (these are behind the CERN SSO)



cmsbuild commented 19 hours ago

cms-sw member

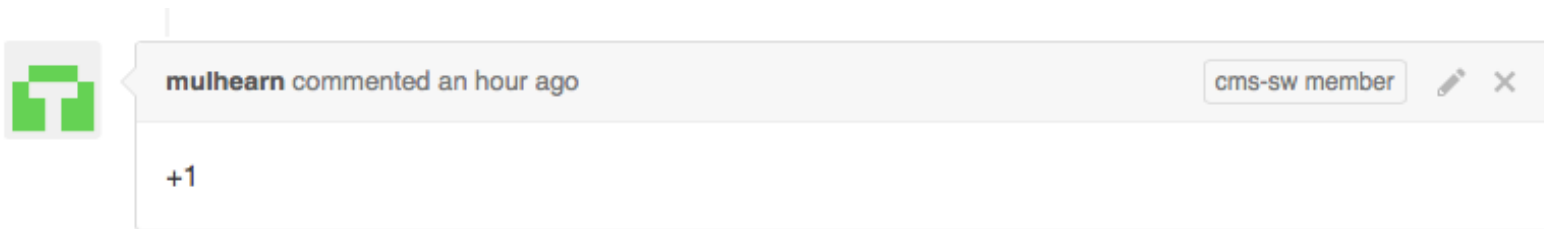


Comparison is ready

<https://cmssdt.cern.ch/SDT/jenkins-artifacts/pull-request-integration/PR-13607/11729/summary.html>

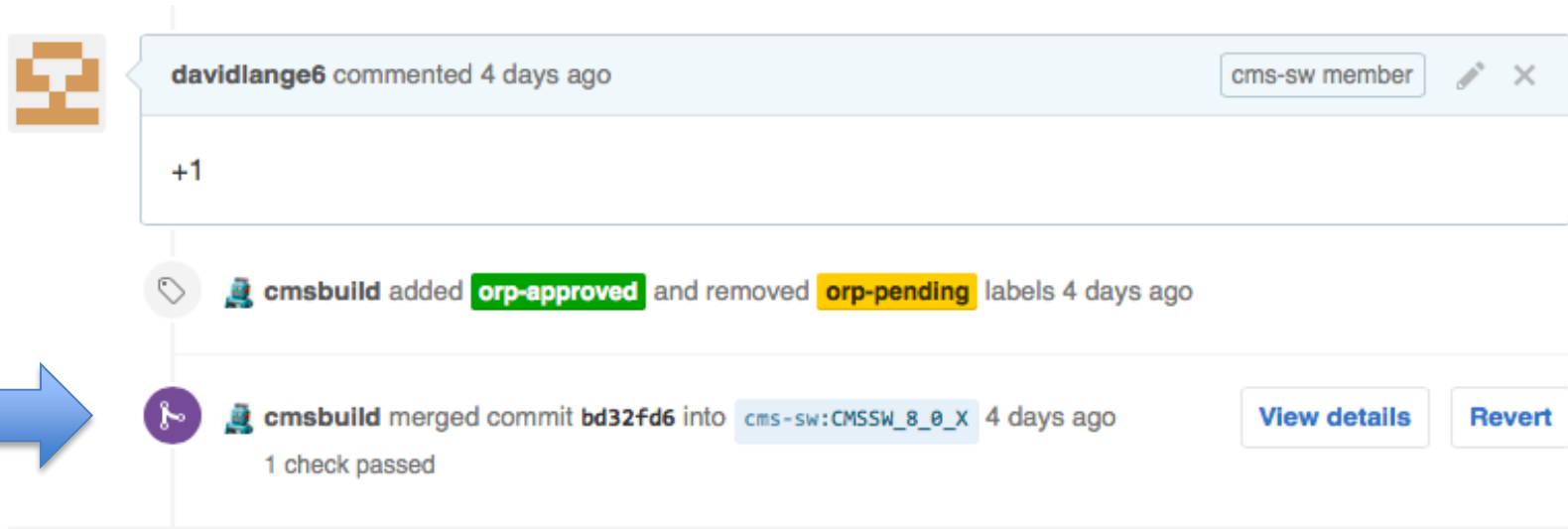
Code request lifecycle (example)

Request is approved by category manager



A GitHub comment by user **mulhearn** (profile picture: green 'T' icon) from the **cms-sw** organization. The comment says "+1" and was posted "an hour ago". It includes a "cms-sw member" label and edit/delete icons.

Request is approved by release manager and integrated into CMSSW



A sequence of GitHub events for a commit:

- A comment by **dauidlange6** (profile picture: orange 'T' icon) from the **cms-sw** organization, saying "+1" and posted "4 days ago".
- An event by **cmsbuild** (robot icon) that "added **orp-approved** and removed **orp-pending** labels 4 days ago".
- The current commit event by **cmsbuild** (robot icon) that "merged commit **bd32fd6** into **cms-sw:CMSSW_8_0_X** 4 days ago". It notes "1 check passed" and includes "View details" and "Revert" buttons.

A large blue arrow points to the final commit event.

Successes / issues

- Git has proven much better for managing complex change requests and has reduced the interference between concurrent requests considerably
- GitHub has proven very reliable (much higher up time percentage than the CERN CVS service had for CMS)
- Despite changing the vision of our workflow after the initial migration, we have an efficient and easy to maintain system for integration and release builds
- We left some users behind (as expected)
- Information private to CMS needs another solution rather than GitHub

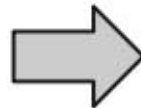
GitHub+Jenkins workflow for releases

- **Jenkins**



- -JAVA based continuous integration system

- **Git and Github**



cms-bot



- <https://github.com/cms-sw/cms-bot>
- Python and shell scripts to **automate our workflows.**
- Self-sentient and very friendly, designed to comply with the 3 laws of robotics.

New Github Issue:

Build <Release Name>



Release is built automatically.



Development Release CMSSW_7_3_0_pre2 now available at CERN

Forum: Software Release Announcements
Date: Nov 11, 17:39
From: David Mandic <David.Mandic>

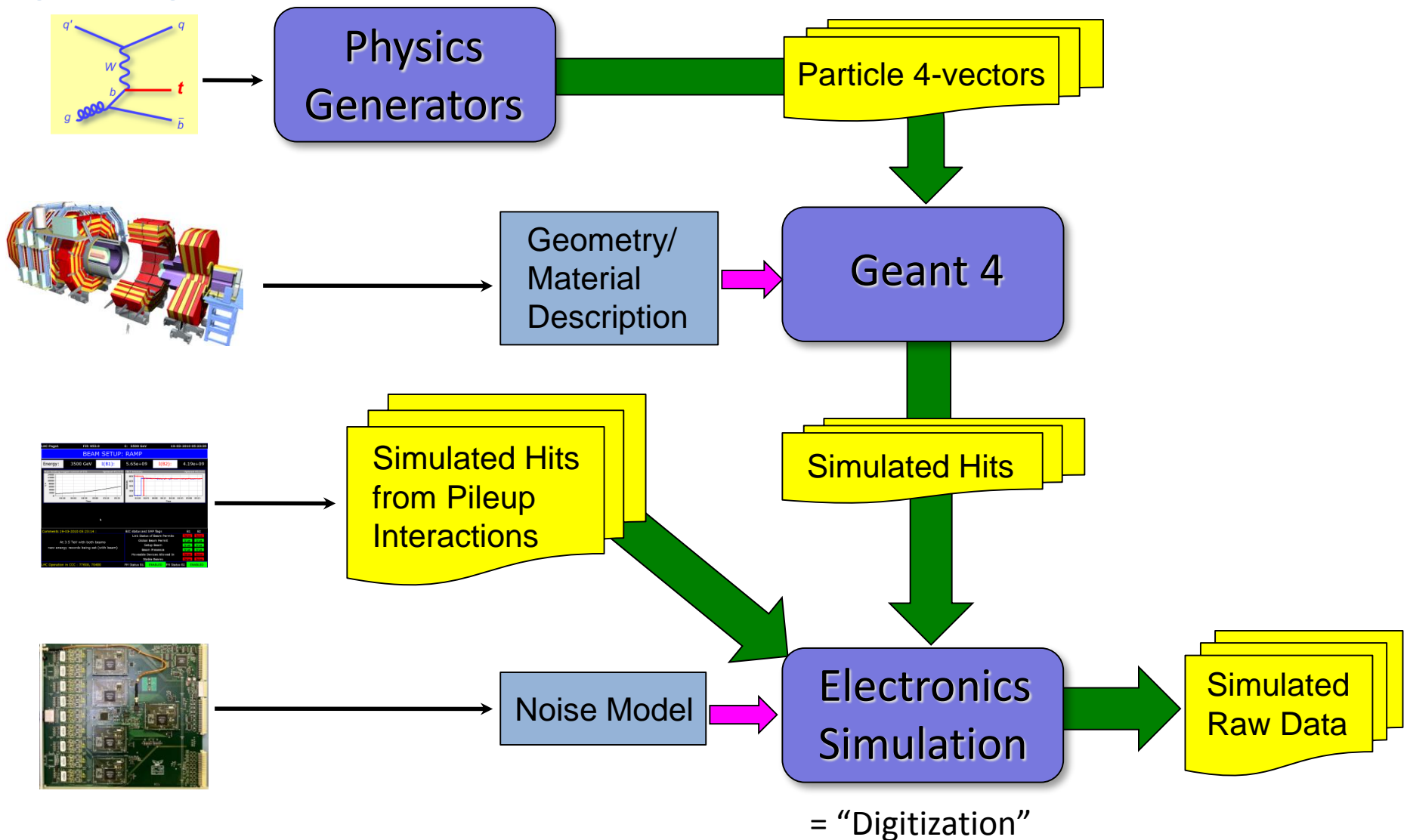
Hi all,
The development release CMSSW_7_3_0_pre2 is now available for the following architectures:
cshw_ami4_gcc61
cshw_ami4_gcc62
The release notes of what changed with respect to CMSSW_7_3_0_pre1 can be found at:
https://github.com/cms-sw/cmssw/blob/master/CMSSW_7_3_0_pre2
Cheers,
David Mandic
! RHC part of type text/text without a line stripped !

Release / integration building system

- In production for ~2 years (100-200 release builds). System supports ~7 active release cycles
 - Build, testing, upload, install steps are each triggered by “+1” from release manager
 - Same infrastructure sits **behind integration build** system (2x per day per release per architecture) and **pull request** testing
 - Means reduced system complexity and IBs provide a testing facility of release build software
 - Straightforward to integrate tests into each build. Tests run vary by type of build
 - Recently expanded to include testing of “external” changes (eg, Pythia8, Geant4 version updates)

SIMULATION

Simulation approach including digitization and pileup simulation



Geant4 status in CMS

- **Production version of Geant4 for 2015-2016**
 - Geant4 version 10.0+patches built in sequential mode
 - Default physics List **QGSP_FTFP_BERT_EML** (Best agreement with CMS test beam data in studies years ago)
 - CMS produced ~9 billion events in 2015
- For 2016: Most CMS simulation samples re-use the detector simulation samples we generated in 2015
 - Typical approach for us when no detector changes are made.
- CMS installs a new pixel detector in 2017, so we will try to update the detector simulation software (Pythia8 tunes, G4, etc)

Geant4 status in CMS – development for 2017

- **Current development version of Geant4 in CMS is Geant4 10.2+patches**
 - Multi-threaded Geant4 is fully integrated with CMS multi-threaded framework
 - Updated physics lists given test beam results currently under evaluation
- Preliminarily: 10.2 shows worse agreement with test beam data. This is under investigation together with the G4 hadronic team
 - Changes to our physics list and patches to 10.2 now under evaluation

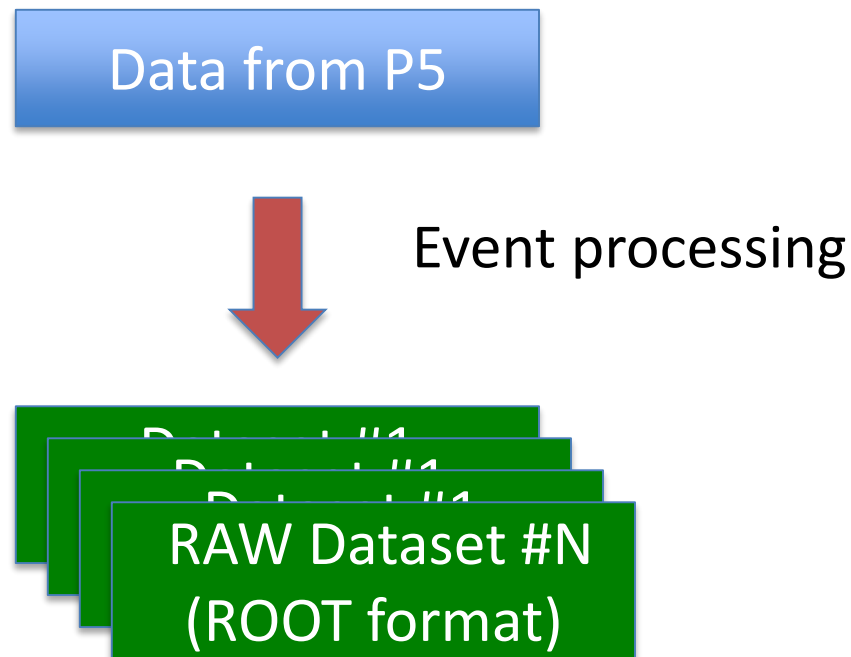
TIER-0 / RECONSTRUCTION CONFIGURATION+WORKFLOWS

Tier-0 workflows and configuration

- Primary evolution during Run 2
 - Multithreaded (typically 4 threads)
 - Added “MiniAOD” output
 - Meant to be small and easily reproducible starting from Run 1 analysis data tier (“AOD”).
 - Multithreading allowed us to add “prompt skims” for physics and detector studies as part of our Tier-0 workflow
 - Previously done on Tier-1 outside of Tier-0 infrastructure

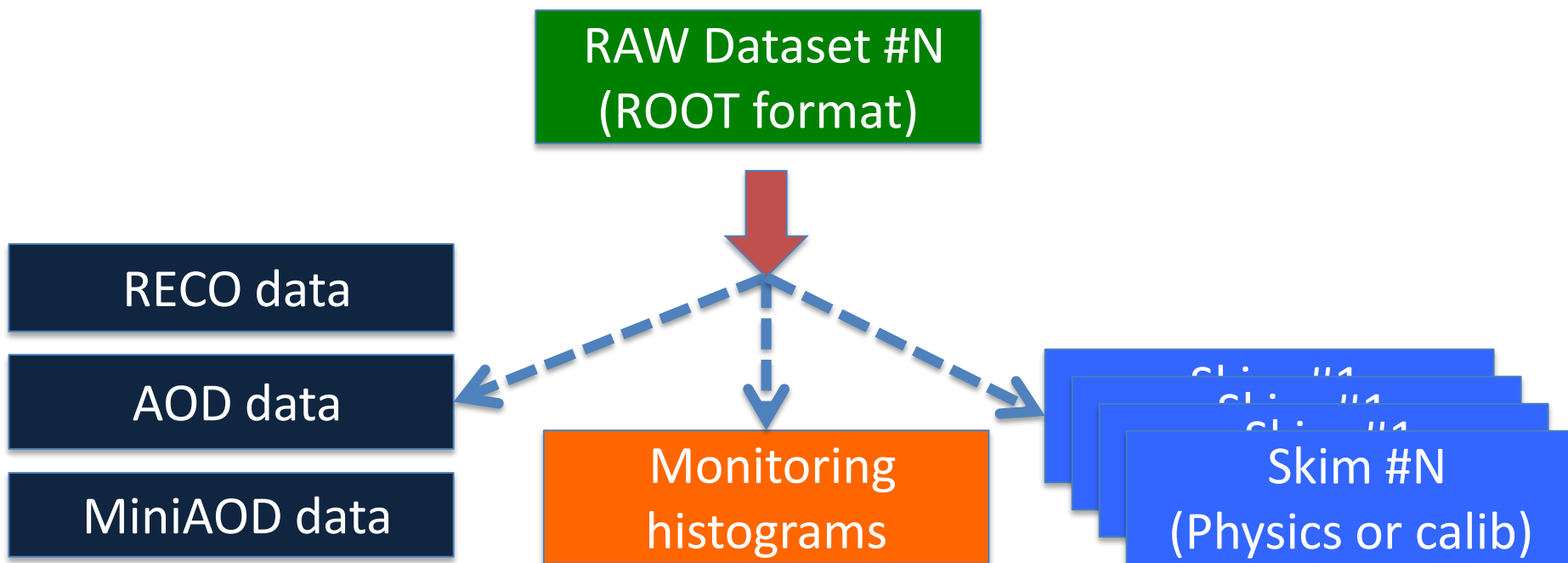
Tier-0 workflows: Repacking step

- Split events into dataset using HLT decision bits and convert to archival RAW data format (ROOT based)



Tier-0 workflows: Reconstruction step

- Perform all event processing in single step
- Today we have only a few skims. We have ideas for how to better isolate individual skim configurations from each other (and rest of application) in case their complexity grows

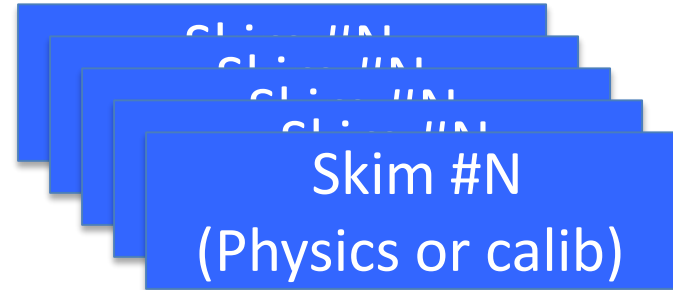


Tier-0 workflows: Merging and Harvesting



Aggregation

Full Run statistics
uploaded to GUI



Aggregation

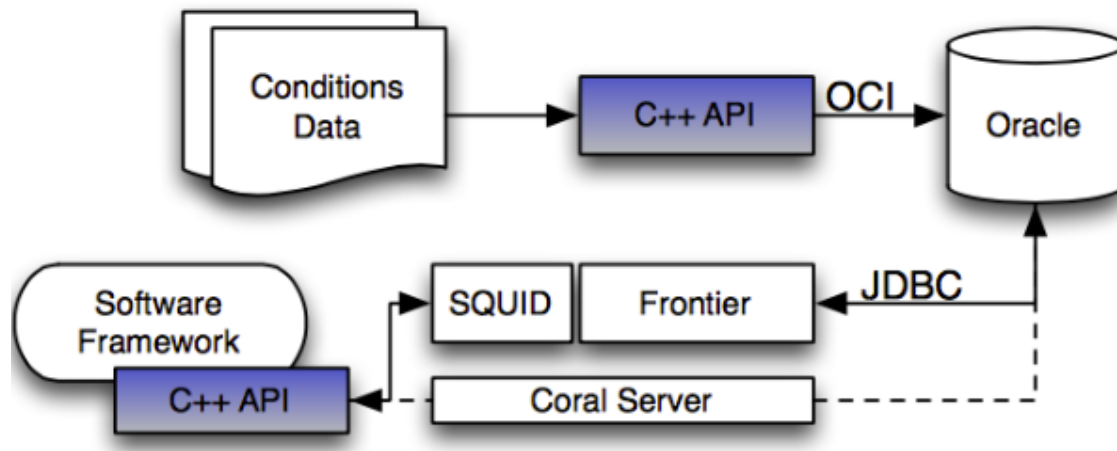
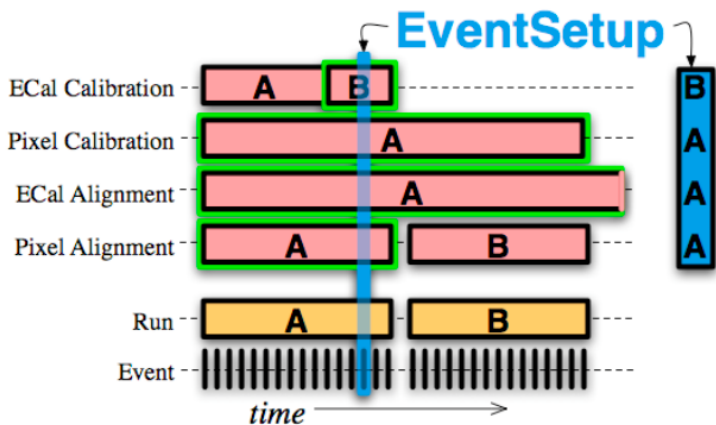
Prompt
Calibration

CONDITIONS

Alignment and Calibration (non-event) data: Run 2 Conditions system in CMS

- Conditions infrastructure rebuilt based on lessons learned during Run 1
 - Reduced complexity of data representation: Multiple tables per conditions object became 1 blob
 - “Global tags” handled in more natural way
- CMS conditions vary with run/lumi (mostly) or time (a few)
 - Multithreaded framework relies on lumi boundaries as the synchronization point

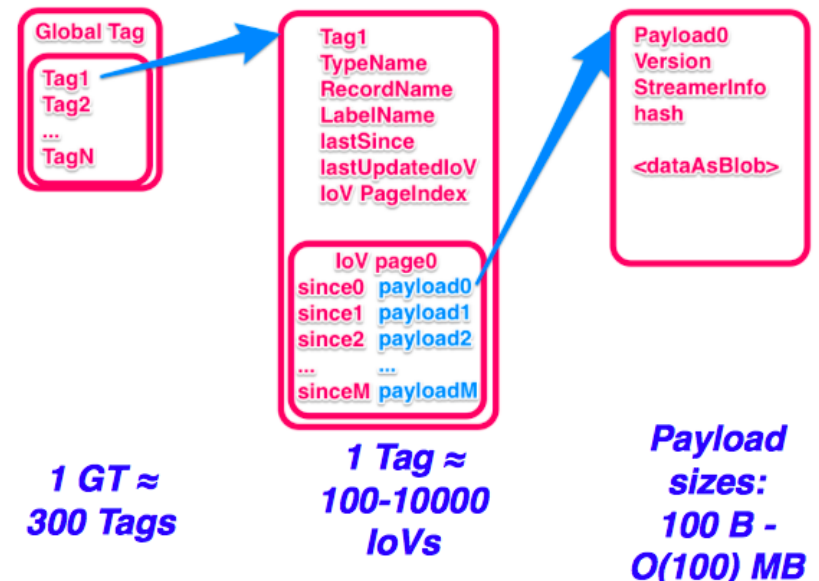
Alignment and Calibration (non-event) data: Run 2 Conditions system in CMS



```
ESHandle<TrackerGeometry> geomPtr;  
eventSetup.get<TrackerAlignmentRecord>()  
    .get( geomPtr );
```

Conditions model

- **Conditions data:** Serialized and stored as **blob** in database
 - We chose to use boost serialization package
- **Interval of validity (IOV):**
 - Defined by “since” (time, lumi) with an open IOV
 - We do not have a use case for very fine grained IOVs. Would require an interface to retrieve “until” (time, lumi) for framework synchronization
- **Global tag:** Defined by a consistent set of tags



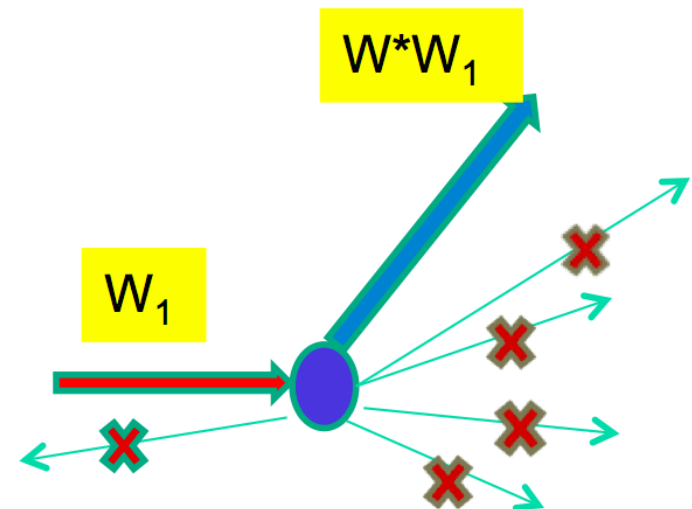
Assessment after one year of operations

- Load on DBAs and experts-on-call is dramatically reduced
- Oracle satisfies our requirement for a highly reliable database service
 - With blob and our IOV schema, DB queries are simple and easy to maintain
 - Now able to investigate other solutions for Oracle functionality for Run 3.
- Schema evolution:
 - So far users have not faced issues with the lack of schema evolution support in the serialization
 - There is however a strong coupling to boost version (lack of “forward” compatibility. Needs to be solved in longer term but not a risk to data taking operations

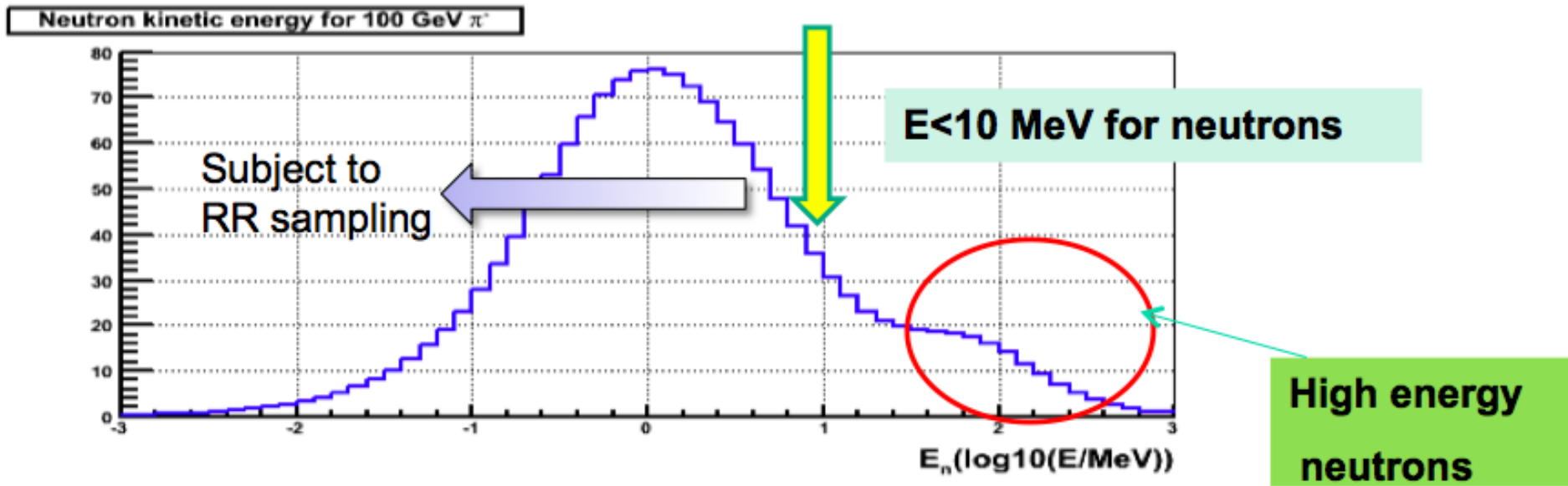
Questions?

Russian Roulette: Sampling of low-energy particles in Geant4

- Method from neutron shielding calculations: Track only a small fraction of low-energy particles through the detector with no noticeable change in simulation results
 - We found that it was necessary to have sampling factors and thresholds that depend on both detector region and particle type.
- Two parameters:
 - RR factor ($1/W$): Fraction of particles to keep
 - Upper energy limit (E_{RR})
- Hits from Particles below E_{RR} that are tracked are given a weight W .



Russian Roulette now used by default after long tuning and validation process



- RR factor of $W=10$ for neutrons and gammas found to give between 25% and 40% performance improvement with no observable effect on physics output
 - Energy and shower shape response in the high-resolution ECAL barrel detector were the most sensitive to RR parameter tuning