The Data Quality Monitoring Software for the CMS experiment at the LHC: past, present and future

Virginia Azzolini\textsuperscript{1},
Broen van Besien\textsuperscript{1},
Dmitrijus Bugelskis\textsuperscript{2},
Tomas Hreus\textsuperscript{3},
Kaori Maeshima\textsuperscript{4},
Javier Fernandez Menendez\textsuperscript{5},
Antanas Norkus\textsuperscript{2},
James Fraser Patrick\textsuperscript{4},
Marco Rovere\textsuperscript{6},
Marcel Andre Schneider\textsuperscript{6}

\textsuperscript{1}Massachusetts Inst. of Technology (US), \textsuperscript{2}Vilnius University (LT), \textsuperscript{3}Universitaet Zuerich (CH), \textsuperscript{4}Fermi National Accelerator Lab. (US), \textsuperscript{5}Universidad de Oviedo (ES), \textsuperscript{6}CERN

on behalf of the CMS collaboration

2018-07-12, CHEP 2018, Sofia, Bulgaria
Introduction

- **Data Quality Monitoring** provides
  1. *monitoring*: detect poor detector performance and malfunctions
  2. *certification*: assess and record quality of data and software releases
  3. *debugging*: provide detailed information in case of problems
Introduction

- **Data Quality Monitoring** provides
  1. *monitoring*: detect poor detector performance and malfunctions
  2. *certification*: assess and record quality of data and software releases
  3. *debugging*: provide detailed information in case of problems

- Intermediate output: *monitor elements* (MEs)
The DQM System

DQM performance
Future Plans

producing monitor elements
visualizing DQM data
taking decisions

The DQM System

CMS

HLT Prompt
Reco
DQM
MC RelVal

HDQM

validators
DC
experts
P5
shift

DQMGUI

DQMGUI

DQMGUI

DQMGUI

run

monitor
elements

“good/bad”

Run Registry

golden JSON

detector
experts

code

DQM

DQM

DQM

DQM

M. Schneider, CHEP 2018
CMS DQM Software: Past, Present and Future
DQM as part of the CMS event processing software (CMSSW)

- DQM code in CMSSW
- Code developed and maintained by subsystem experts
- **DQM framework** to save MEs (non-event data) derived from event data
The online DQM system

- Dedicated machines at the experiment site run DQM code
- Independent processes per subsystem, very flexible
- Also provides event display and online beamspot measurement
The DQMGUIs

- Aggregate and archive MEs from CMSSW
- Provide APIs for custom tools/scripts
The DQM System
DQM performance
Future Plans
producing monitor elements
visualizing DQM data
taking decisions

The DQMGUIs

- Aggregate and archive MEs from CMSSW
- Provide APIs for custom tools/scripts
- Provide a web-based UI to browse and view data
Historic DQM

- Provides a time-based view of DQM data

---

- The DQM System
- DQM performance
- Future Plans

- producing monitor elements
- visualizing DQM data
- taking decisions
Historic DQM

▶ Provides a time-based view of DQM data
▶ New central DQM tool, originally built by subsystem developers
▶ Only for debugging, not needed for monitoring and certification
Humans are a central part of DQM

All CMS data is centrally checked and certified for analysis
Run Registry

- CMS
- HLT
- Prompt Reco
- MC
- RelVal
- DQM
- DQMGUI
- HDQM
- DQMGUI
- P5 shift
- DC experts
- detector experts

- online
- offline
- simulation
- event data
- monitor elements
- "good/bad"

- golden JSON
- validated release
- Run Registry

- code

- producing monitor elements
- visualizing DQM data
- taking decisions

Run Registry

M. Schneider, CHEP 2018

CMS DQM Software: Past, Present and Future

9 / 13
The DQM System
DQM performance
Future Plans
producing monitor elements
visualizing DQM data
taking decisions

Run Registry

▶ Web service to collect certification results
▶ APIs for scripts to produce the final list of data ready for analysis (golden JSON)

CMS DQM Run Registry (GLOBAL)

M. Schneider, CHEP 2018
### Run Registry

#### Web service to collect certification results
- ▶ Web service to collect certification results
- ▶ APIs for scripts to produce the final list of data ready for analysis (golden JSON)
- ▶ Currently under redesign for better usability and maintainability

<table>
<thead>
<tr>
<th>Run Registry</th>
<th>ONLINE</th>
<th>OFFLINE</th>
<th>USER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WORKSPACES</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GLOBAL</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number</th>
<th>LHC FI</th>
<th>B1 stable</th>
<th>B2 stable</th>
<th>B-Field</th>
<th>Events</th>
<th>Started</th>
<th>Stop</th>
</tr>
</thead>
<tbody>
<tr>
<td>317068</td>
<td>6737</td>
<td>0</td>
<td>0</td>
<td>3.0057459375</td>
<td>50663</td>
<td>Mon, 28 May...</td>
<td></td>
</tr>
<tr>
<td>317067</td>
<td>6737</td>
<td>0</td>
<td>0</td>
<td>3.0057459375</td>
<td>Mon, 28 May...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>317066</td>
<td>6737</td>
<td>0</td>
<td>0</td>
<td>3.0057459375</td>
<td>64725</td>
<td>Mon, 28 May...</td>
<td></td>
</tr>
<tr>
<td>317065</td>
<td>6737</td>
<td>0</td>
<td>0</td>
<td>3.0057459375</td>
<td>206457</td>
<td>Mon, 28 May...</td>
<td></td>
</tr>
<tr>
<td>317064</td>
<td>6737</td>
<td>0</td>
<td>0</td>
<td>3.0057459375</td>
<td>5841</td>
<td>Mon, 28 May...</td>
<td></td>
</tr>
<tr>
<td>317063</td>
<td>6737</td>
<td>0</td>
<td>0</td>
<td>3.0057459375</td>
<td>Mon, 28 May...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>317062</td>
<td>6737</td>
<td>0</td>
<td>0</td>
<td>3.0057459375</td>
<td>Mon, 28 May...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>317061</td>
<td>6737</td>
<td>0</td>
<td>0</td>
<td>3.0057459375</td>
<td>Mon, 28 May...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>317059</td>
<td>6736</td>
<td>0</td>
<td>0</td>
<td>3.0057459375</td>
<td>340738</td>
<td>Mon, 28 May...</td>
<td></td>
</tr>
<tr>
<td>317058</td>
<td>6736</td>
<td>0</td>
<td>0</td>
<td>3.0057459375</td>
<td>Mon, 28 May...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>317057</td>
<td>6736</td>
<td>0</td>
<td>0</td>
<td>3.0057459375</td>
<td>Mon, 28 May...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>317056</td>
<td>6736</td>
<td>0</td>
<td>0</td>
<td>3.0057459375</td>
<td>Mon, 28 May...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>317055</td>
<td>6736</td>
<td>0</td>
<td>0</td>
<td>3.0057459375</td>
<td>Mon, 28 May...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>317054</td>
<td>6736</td>
<td>0</td>
<td>0</td>
<td>3.0057459375</td>
<td>Mon, 28 May...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>317053</td>
<td>6736</td>
<td>0</td>
<td>0</td>
<td>3.0057459375</td>
<td>643073</td>
<td>Mon, 28 May...</td>
<td></td>
</tr>
<tr>
<td>317052</td>
<td>6736</td>
<td>0</td>
<td>0</td>
<td>3.0057459375</td>
<td>271464</td>
<td>Mon, 28 May...</td>
<td></td>
</tr>
<tr>
<td>317051</td>
<td>6736</td>
<td>0</td>
<td>0</td>
<td>3.0057459375</td>
<td>8320</td>
<td>Mon, 28 May...</td>
<td></td>
</tr>
<tr>
<td>317049</td>
<td>6736</td>
<td>0</td>
<td>0</td>
<td>3.0057459375</td>
<td>3590</td>
<td>Mon, 28 May...</td>
<td></td>
</tr>
<tr>
<td>317048</td>
<td>6736</td>
<td>0</td>
<td>0</td>
<td>3.0057459375</td>
<td>9</td>
<td>Mon, 28 May...</td>
<td></td>
</tr>
</tbody>
</table>

- BTAG
- CASTOR
- CSC
- CTPPS
- DT
- ECAL
- EGAMMA
- HCAL
- HLT
- JETMET
- LIT

M. Schneider, CHEP 2018
Some statistics

- The DQMGUIs provide access to
  - Online: 22,000 runs, 650GB of data
  - Offline: 400,000 datasets, 4100GB of data
Some statistics

- The DQMGUIs provide access to
  - Online: 22,000 runs 650GB of data
  - Offline: 400,000 datasets 4100GB of data

- $4.4 \cdot 10^{10}$ MEs in the offline DQMGUI, covering $3.8 \cdot 10^{10}$ events processed by DQM
Some statistics

- The DQMGUIs provide access to
  - Online: 22,000 runs 650GB of data
  - Offline: 400,000 datasets 4100GB of data

- $4.4 \cdot 10^{10}$ MEs in the offline DQMGUI, covering $3.8 \cdot 10^{10}$ events processed by DQM

- In CMSSW, DQM received 600 pull requests (out of 2000 total) over the last year, plus 100 pull requests for DQMGUI display changes
The memory challenge

- MEs need to be kept in memory, and there are *lots* of them
- CMSSW uses multithreading (vs. jobs) to improve compute resource utilization
- Options for DQM
  1. ensure MEs are only accessed by a single thread
  2. make ME updates thread-safe (atomic)
  3. duplicate MEs and reduce to final result
The memory challenge

- MEs need to be kept in memory, and there are *lots* of them
- CMSSW uses multithreading (vs. jobs) to improve compute resource utilization
- Options for DQM
  1. ensure MEs are only accessed by a single thread
  2. make ME updates thread-safe (atomic)
  3. duplicate MEs and reduce to final result
- 2015: use solution 3. (high memory usage!)
The memory challenge

- MEs need to be kept in memory, and there are *lots* of them
- CMSSW uses multithreading (vs. jobs) to improve compute resource utilization
- Options for DQM
  1. ensure MEs are only accessed by a single thread
  2. make ME updates thread-safe (atomic)
  3. duplicate MEs and reduce to final result
- 2015: use solution 3. (high memory usage!)
  2017: running out of memory, convert some MEs to 2.
The memory challenge

- MEs need to be kept in memory, and there are *lots* of them
- CMSSW uses multithreading (vs. jobs) to improve compute resource utilization
- Options for DQM
  1. ensure MEs are only accessed by a single thread
  2. make ME updates thread-safe (atomic)
  3. duplicate MEs and reduce to final result
- 2015: use solution 3. (high memory usage!)
- 2017: running out of memory, convert some MEs to 2.
- 2018: convert the remaining MEs to 1.
The memory challenge

- MEs need to be kept in memory, and there are *lots* of them
- CMSSW uses multithreading (vs. jobs) to improve compute resource utilization
- Options for DQM
  1. ensure MEs are only accessed by a single thread
  2. make ME updates thread-safe (atomic)
  3. duplicate MEs and reduce to final result
- 2015: use solution 3. (high memory usage!)
  2017: running out of memory, convert some MEs to 2.
  2018: convert the remaining MEs to 1.
- 1. is fine, since DQM uses little CPU time and plugins accessing independent MEs can still run concurrently.
Future Plans

- Cleanup and disentangle the DQM configuration
  - Remove MEs that are no longer needed
  - Clearly document which modules are used for which purposes
  - Allow fine-grained enabling/disabling of code without breaking dependencies
Future Plans

- Cleanup and disentangle the DQM configuration
  - Remove MEs that are no longer needed
  - Clearly document which modules are used for which purposes
  - Allow fine-grained enabling/disabling of code without breaking dependencies

- Improve the data certification process
  - New Run Registry under construction, to improve flexibility and maintainability
  - Integrate machine learning based feedback (see Adrian’s talk in Track 6)
Future Plans

- Cleanup and disentangle the DQM configuration
  - Remove MEs that are no longer needed
  - Clearly document which modules are used for which purposes
  - Allow fine-grained enabling/disabling of code without breaking dependencies
- Improve the data certification process
  - New Run Registry under construction, to improve flexibility and maintainability
  - Integrate *machine learning* based feedback (see Adrian’s talk in Track 6)
- Improve tooling around DQMGUI
  - Provide well-defined and performant APIs
  - Make data more accessible, maybe interactive notebooks (Jupyter/SWAN)
Future Plans

- Cleanup and disentangle the DQM configuration
  - Remove MEs that are no longer needed
  - Clearly document which modules are used for which purposes
  - Allow fine-grained enabling/disabling of code without breaking dependencies

- Improve the data certification process
  - New Run Registry under construction, to improve flexibility and maintainability
  - Integrate machine learning based feedback (see Adrian’s talk in Track 6)

- Improve tooling around DQMGUI
  - Provide well-defined and performant APIs
  - Make data more accessible, maybe interactive notebooks (Jupyter/SWAN)?

- Prepare for more data
We are quite happy with the Run2 DQM system
Conclusion

- We are quite happy with the Run2 DQM system
- But there are things to improve
  - Mostly for maintainability after a few years of incremental changes
  - But also to make DQM more useful for the experiment