ATLAS Release Tester (ART)

Tülay Çuhadar Dönszelmann (UCI)

IRIS-HEP topical meetings on US-ATLAS and US-CMS
March 29th, 2021

Introduction

- ATLAS Offline Software Release
  - — Lives in one large git repository
  - — One branch for each release-series (Tier0-production, simulation, development)
  - — Code-base is sub-divided into “*packages*” (feature of our build system)
  - — *Projects* can be defined as subset of “packages”
    - Example: Event Generation releases use only a subset of packages
- Nightly Builds
  - — HEAD of each branch
  - — Tagged for each nightly by timestamp
  - — Multiple platforms and multiple projects
- Nightly Tests
  - — Run for each of the nightly builds
  - — Short Tests (executed locally)
  - — Long Tests (executed on Grid)

Previous Nightly Testing System

- Run Time Tester (RTT) framework has been used in ATLAS for a long time
  - The system was bound to specific clusters at CERN
  - Depended on AFS
  - Running a single test was not straightforward, because all tests were defined in a single XML file

A new framework for the ATLAS testing system was needed to address these issues

What is ART ?

- ATLAS Release Tester (ART) provides a unified testing system
  - One tool (art command line)
  - One set of tests (for grid or local)
- It allows to submit:
  - Long tests to the GRID
  - Short tests on local machines, to be run in parallel
- It is used by:
  - Automatic Nightly Submission
    - After the nightly release is built
    - Using the gitlab-ci system to manage the submission
  - Users
    - To run jobs locally or on the GRID

Features of ART

- Simple Test Definition
  - Shell or Python tests
  - Adorned with *headers* to instruct ART
  - Full control by developers
  - Easy to run and reproduce any failure
  - Easy to submit job to GRID
- Predefined set of possible input files (bytestream, simulation, ...)
  - Either on *CVMFS* or on *GRID* (rucio)
- Possibility to run post processing
  - Regression tests
  - Histogram comparison
- Automatic download and storage of results
  - Automatic clean up the EOS

ART Command Line Utilities

- User defines test and adds art-headers in the form of key-value pairs:

```
test_example.sh(.py)
# art-type: grid
# art-input: …
…

<actual test lines go here>
```

# art-type : grid | local	(To run on grid or locally)
# art-include: <String>	(Nightlies the script must run on)
# art-input: <String>	(Name of the dataset to be read in the grid)
# art-nfiles: <Int>	(Number of files to be read from the dataset)

- User run jobs in parallel locally or submits jobs to GRID using ART

```
art.py run       [options] <script_directory> <sequence_tag> [<test_names>…]
art.py grid      [options] <script_directory> <sequence_tag>
```

waits for grid result to be ready to copy to EOS using ART:

```
art.py copy       <indexed_package>
```

- Some ART Command line utilities (CLU):

```
ART - ATLAS Release Tester.


Usage:
  art.py run             [-v -q --type=<T> --max-jobs=<N> --ci --run-all-tests --timeout=<S> --copy=<dir>]
<script_directory> <sequence_tag> [<test_names>...]
  art.py grid            [-v -q --type=<T> --max-jobs=<N> -n --run-all-tests --no-build=<nightly_tag>] <script_directory>
<sequence_tag>
  art.py submit          [-v -q --type=<T> --max-jobs=<N> --config=<file> -n --run-all-tests --no-build=<nightly_tag> --
no-copy] <sequence_tag> [<packages>...]
  art.py copy            [-v -q --user=<user> --dst=<dir> --unpack --tmp=<dir> --seq=<N> --keep-tmp] <indexed_package>
  art.py validate        [-v -q] [<script_directory>]
  art.py included        [-v -q --type=<T> --test-type=<TT> --out=<file>] [<script_directory> [<packages>...]]
```

ART Implementation

- Simple class hierarchy to handle local and grid jobs

  — Fully written in python

- Helper classes to abstract different functionality for things such as configuration, headers, Rucio:

  — *ArtConfiguration, ArtHeader, ArtRucio*

- Some scripts to handle different functionality:

  — *art.py* (main script), *art-trigger.py* (sending trigger to git-lab-ci), *art-share.py* (input management)

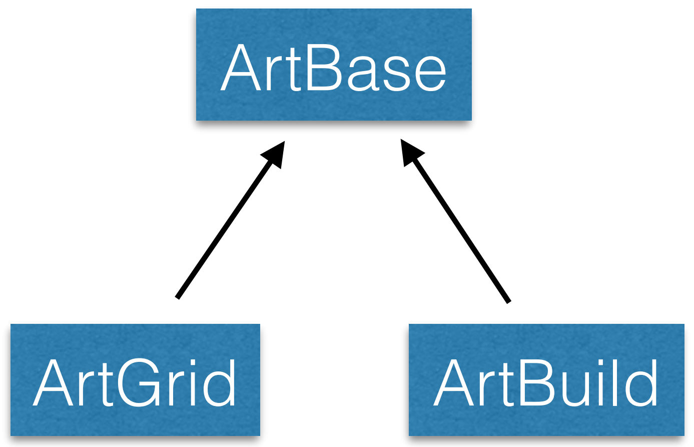
- ART is on gitlab https://gitlab.cern.ch/art in four projects:

  art-sw: ART software project, Classes and command-line tool

  art-submit: ART grid submission project, receiving the trigger and submitting the jobs

  art-gitlab-ci-runner: Runner images (slc6, cc7, grid and local) for ART

  art-www: ART project web site and asciidoc manual



Automatic Nightly Submission

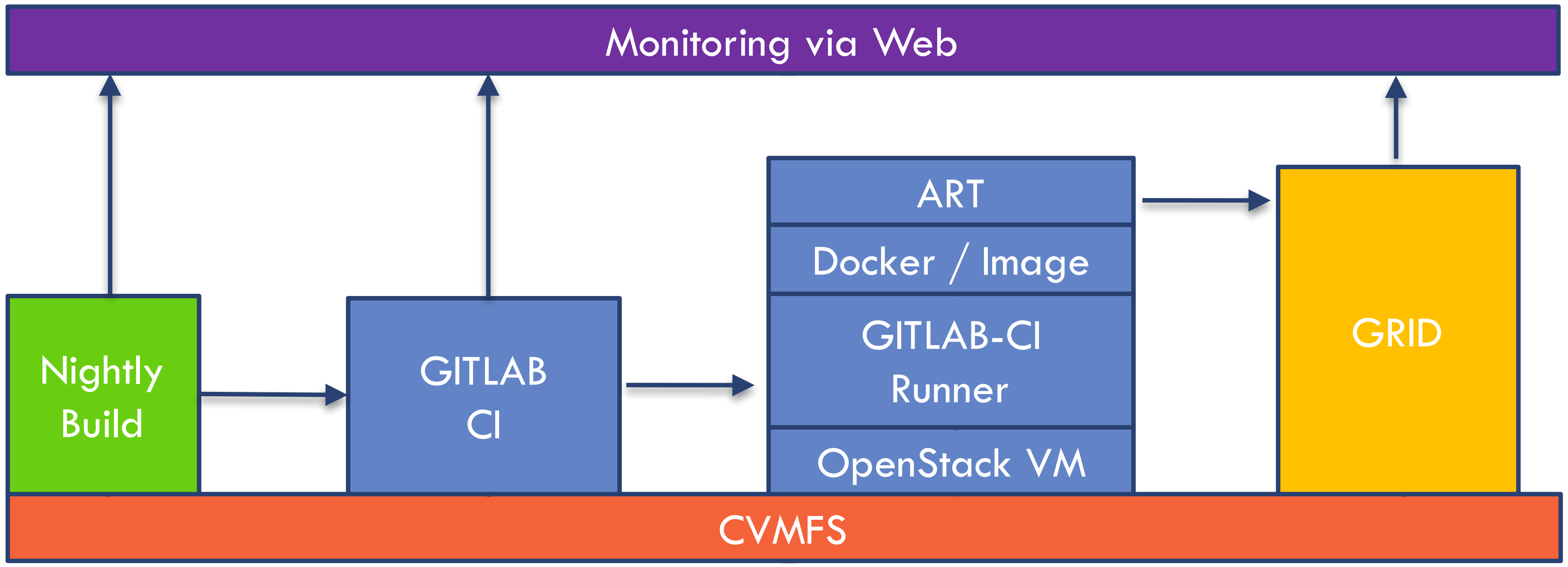
- Nightly Build triggers the ART gitlab-ci system, which runs through 4 stages:

  checkout: Checks out a proper copy of ART

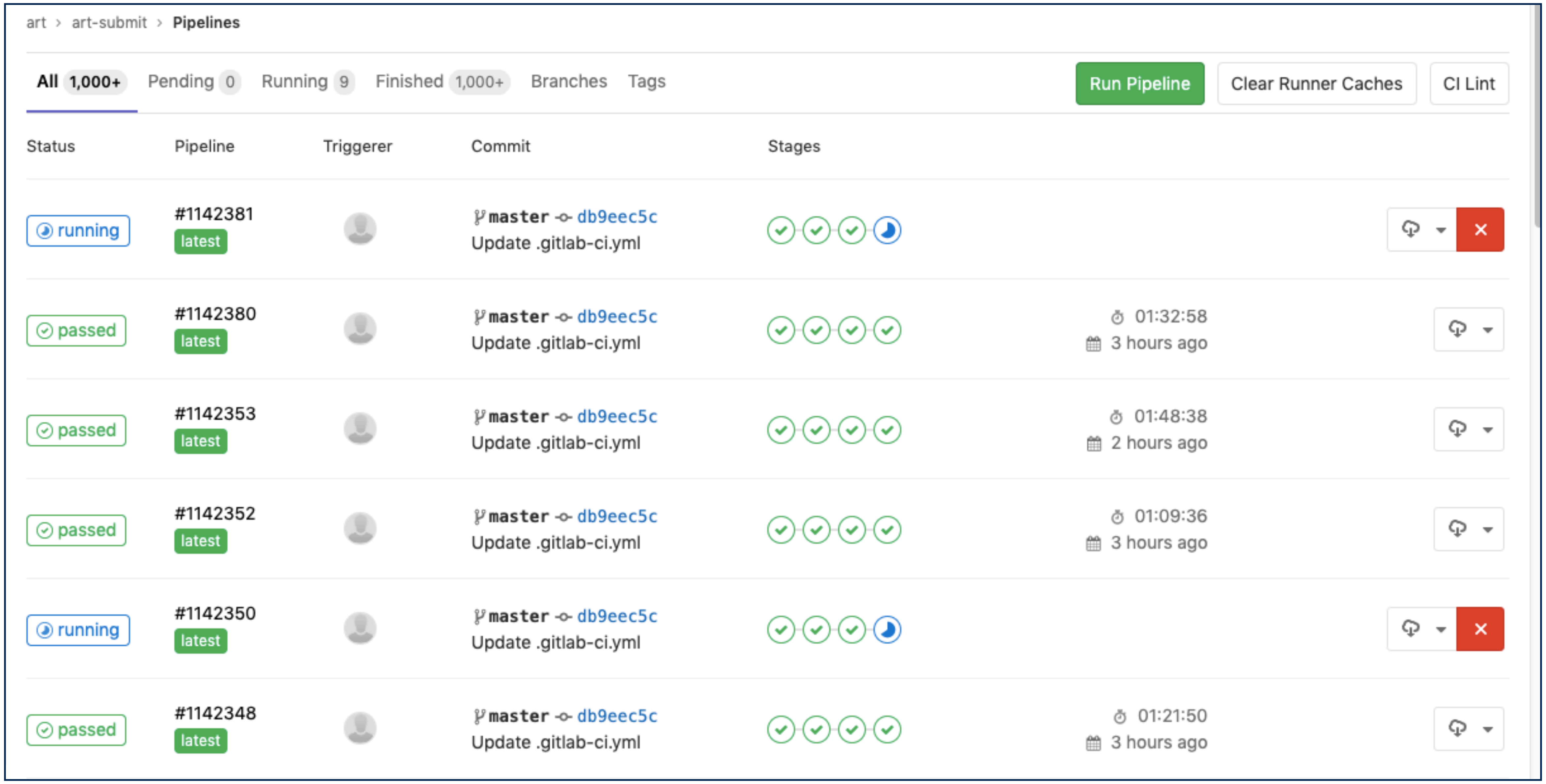
  configure: Verifies if testing is required

  cvmfs: Verifies the availability of the nightly release on CVMFS (which is distributed to the GRID)

  submit: Submits jobs to grid (ART CLU) and waits for results to be copied
- The 4 stages above run on a set of 5 Virtual Machines for ART, each loaded with docker images to run the ART command line and submit jobs to the grid or to the two machines dedicated for ART local jobs.
- Jobs can be consulted using a Web Interface looking at either gitlab or GRID output.



art-submit Pipeline


art › art-submit › Pipelines

All 1,000+ | Pending 0 | Running 9 | Finished 1,000+ | Branches | Tags

Run Pipeline | Clear Runner Caches | CI Lint

Status	Pipeline	Triggerer	Commit	Stages		
running	#1142381 latest		master db9eec5c Update .gitlab-ci.yml	✓ ✓ ✓ ◔		
passed	#1142380 latest		master db9eec5c Update .gitlab-ci.yml	✓ ✓ ✓ ✓	01:32:58 3 hours ago	
passed	#1142353 latest		master db9eec5c Update .gitlab-ci.yml	✓ ✓ ✓ ✓	01:48:38 2 hours ago	
passed	#1142352 latest		master db9eec5c Update .gitlab-ci.yml	✓ ✓ ✓ ✓	01:09:36 3 hours ago	
running	#1142350 latest		master db9eec5c Update .gitlab-ci.yml	✓ ✓ ✓ ◔		
passed	#1142348 latest		master db9eec5c Update .gitlab-ci.yml	✓ ✓ ✓ ✓	01:21:50 3 hours ago	

Each job corresponds to submission for a nightly

ART's own Continuous Integration (CI)

- Unit and Integration Tests for ART try to cover all its code, runs in gitlab-ci in three phases at every commit.

  Checks: ATLAS setup, python-flake8, shell check (30 seconds)

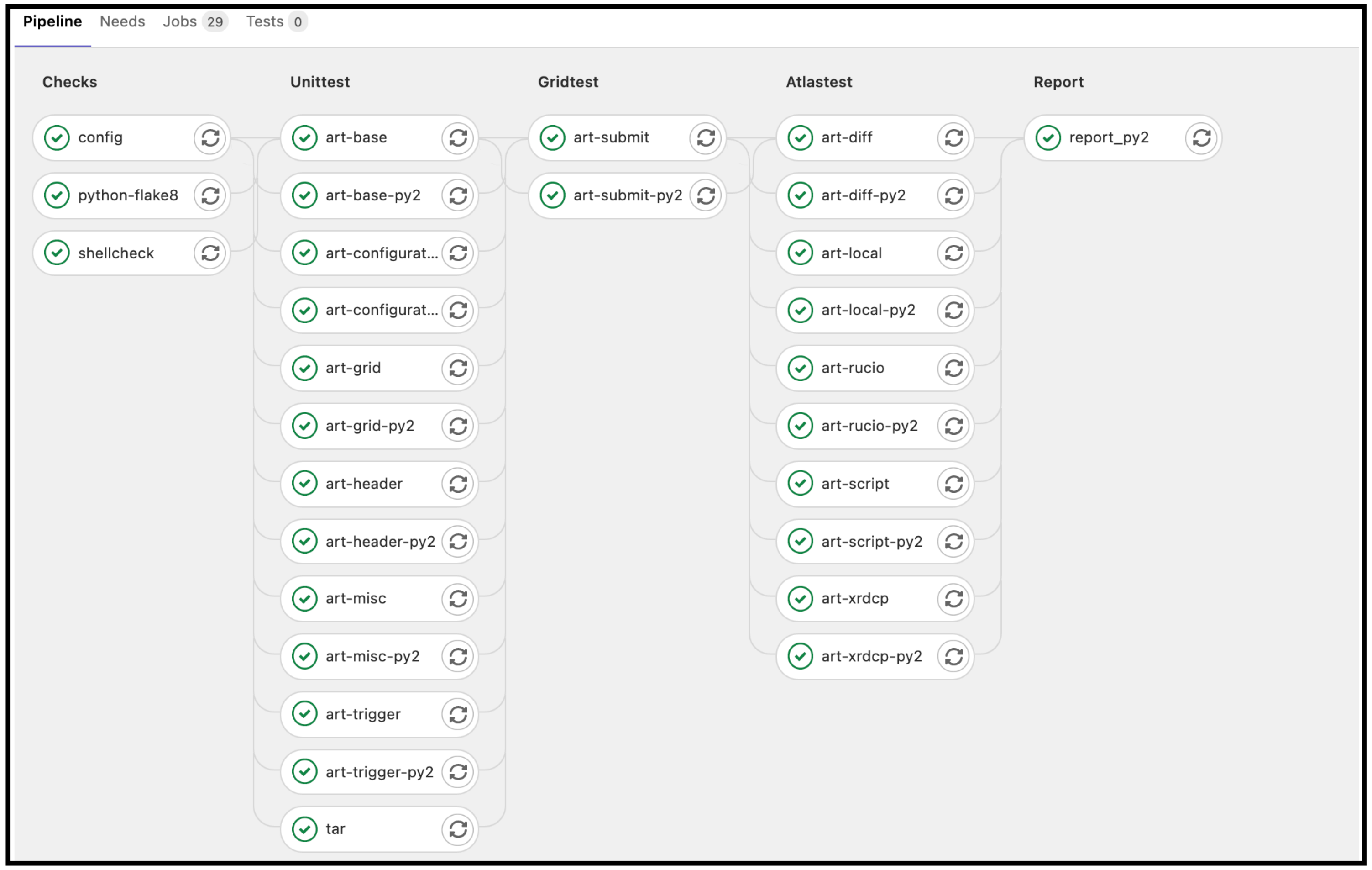
  Unittests: For each of the classes/modules (2 min)

  Atlastests: Local tests to setup and download files (10 min)

  Gridtests: Run when repo is tagged: submit simple job and check results (30 min)

- Coverage: gather all coverage information of unittests and grid-tests and publish

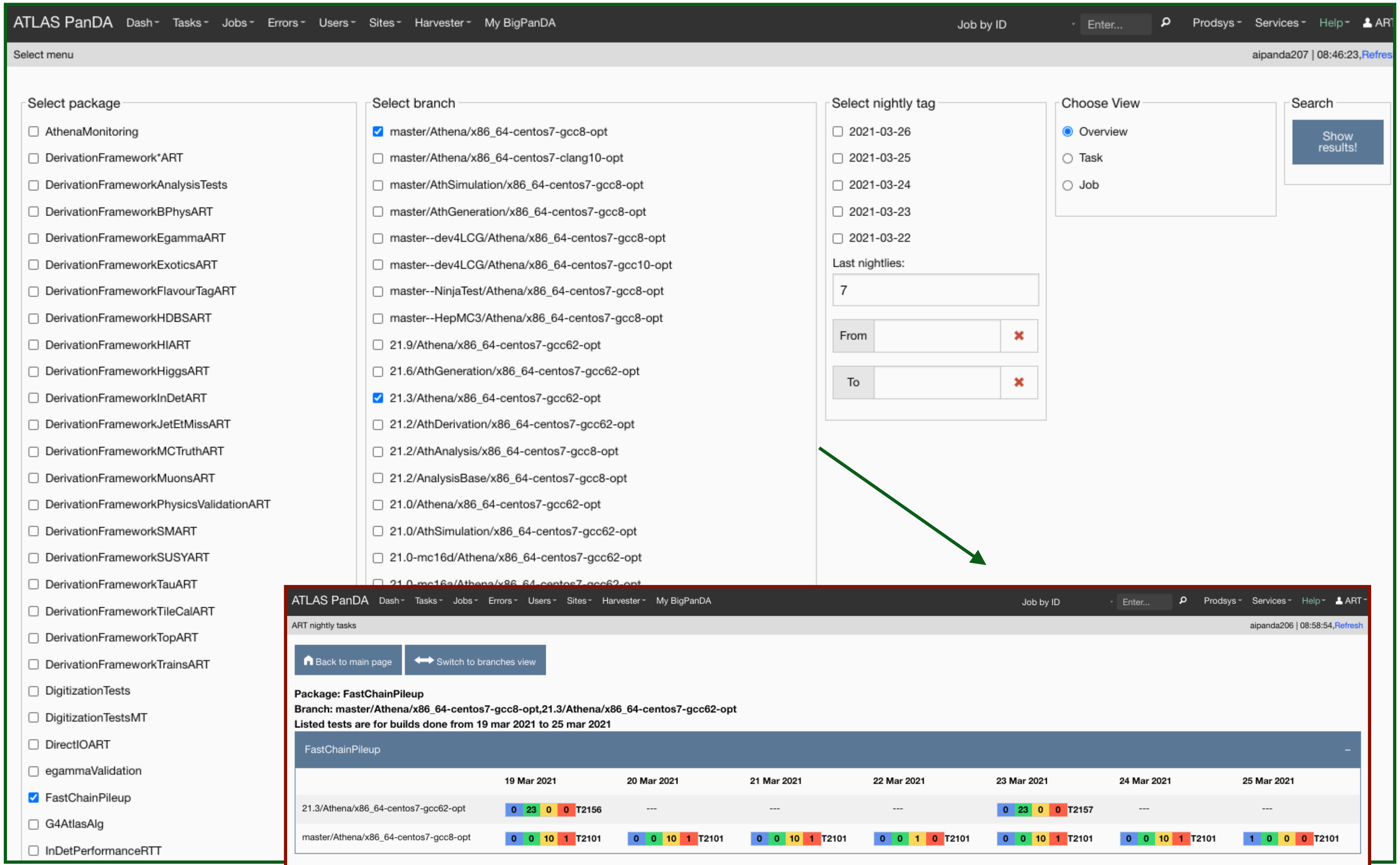
  ➢ Coverage of the code is around 90%



Checking the ART grid jobs

https://bigpanda.cern.ch/art/

BigPanDAmon team (Contact: tatiana.korchuganova@cern.ch)



Select package

- [ ] AthenaMonitoring
- [ ] DerivationFramework*ART
- [ ] DerivationFrameworkAnalysisTests
- [ ] DerivationFrameworkBPhysART
- [ ] DerivationFrameworkEgammaART
- [ ] DerivationFrameworkExoticsART
- [ ] DerivationFrameworkFlavourTagART
- [ ] DerivationFrameworkHDBSART
- [ ] DerivationFrameworkHIART
- [ ] DerivationFrameworkHiggsART
- [ ] DerivationFrameworkInDetART
- [ ] DerivationFrameworkJetEtMissART
- [ ] DerivationFrameworkMCTruthART
- [ ] DerivationFrameworkMuonsART
- [ ] DerivationFrameworkPhysicsValidationART
- [ ] DerivationFrameworkSMART
- [ ] DerivationFrameworkSUSYART
- [ ] DerivationFrameworkTauART
- [ ] DerivationFrameworkTileCalART
- [ ] DerivationFrameworkTopART
- [ ] DerivationFrameworkTrainsART
- [ ] DigitizationTests
- [ ] DigitizationTestsMT
- [ ] DirectIOART
- [ ] egammaValidation
- [✓] FastChainPileup
- [ ] G4AtlasAlg
- [ ] InDetPerformanceRTT

Select branch

- [✓] master/Athena/x86_64-centos7-gcc8-opt
- [ ] master/Athena/x86_64-centos7-clang10-opt
- [ ] master/AthSimulation/x86_64-centos7-gcc8-opt
- [ ] master/AthGeneration/x86_64-centos7-gcc8-opt
- [ ] master--dev4LCG/Athena/x86_64-centos7-gcc8-opt
- [ ] master--dev4LCG/Athena/x86_64-centos7-gcc10-opt
- [ ] master--NinjaTest/Athena/x86_64-centos7-gcc8-opt
- [ ] master--HepMC3/Athena/x86_64-centos7-gcc8-opt
- [ ] 21.9/Athena/x86_64-centos7-gcc62-opt
- [ ] 21.6/AthGeneration/x86_64-centos7-gcc62-opt
- [✓] 21.3/Athena/x86_64-centos7-gcc62-opt
- [ ] 21.2/AthDerivation/x86_64-centos7-gcc62-opt
- [ ] 21.2/AthAnalysis/x86_64-centos7-gcc8-opt
- [ ] 21.2/AnalysisBase/x86_64-centos7-gcc8-opt
- [ ] 21.0/Athena/x86_64-centos7-gcc62-opt
- [ ] 21.0/AthSimulation/x86_64-centos7-gcc62-opt
- [ ] 21.0-mc16d/Athena/x86_64-centos7-gcc62-opt

Select nightly tag

- [ ] 2021-03-26
- [ ] 2021-03-25
- [ ] 2021-03-24
- [ ] 2021-03-23
- [ ] 2021-03-22

Last nightlies: 7

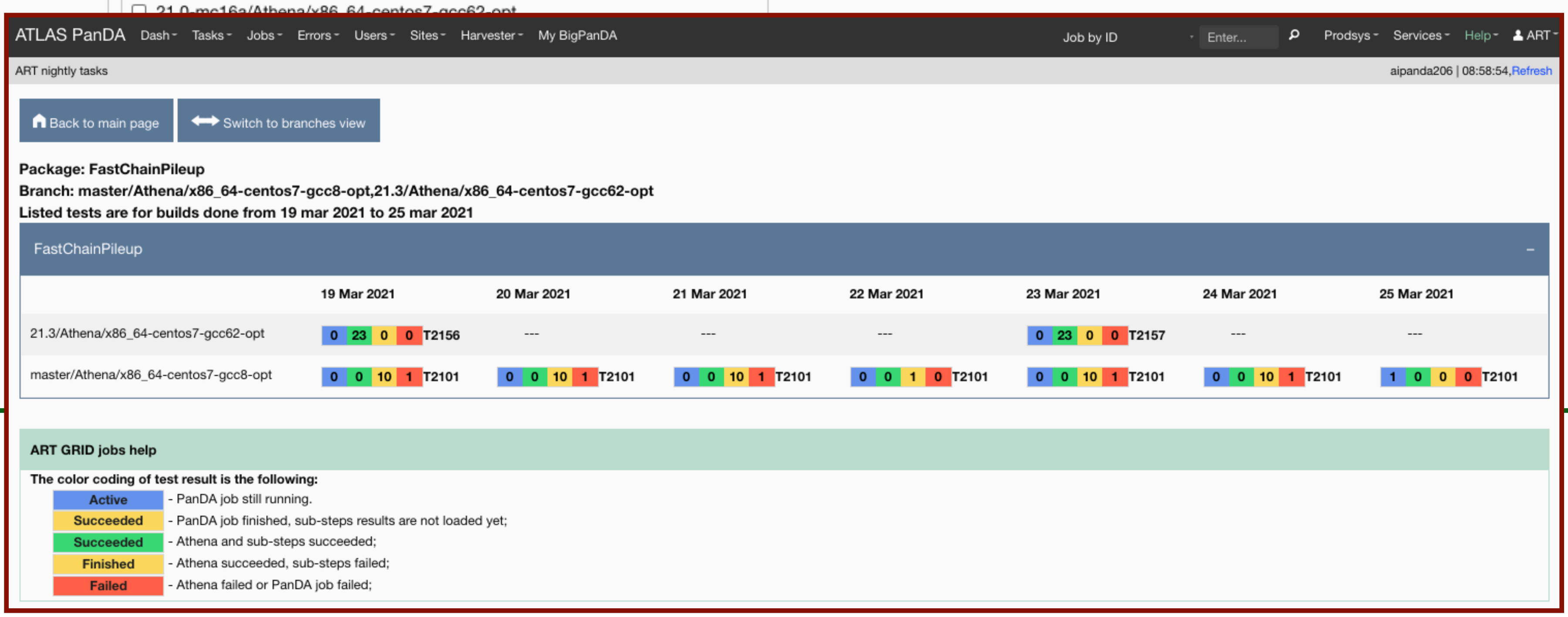
From

To

Choose View

- [● Overview]
- ○ Task
- ○ Job

Search



Package: FastChainPileup
Branch: master/Athena/x86_64-centos7-gcc8-opt,21.3/Athena/x86_64-centos7-gcc62-opt
Listed tests are for builds done from 19 mar 2021 to 25 mar 2021

FastChainPileup

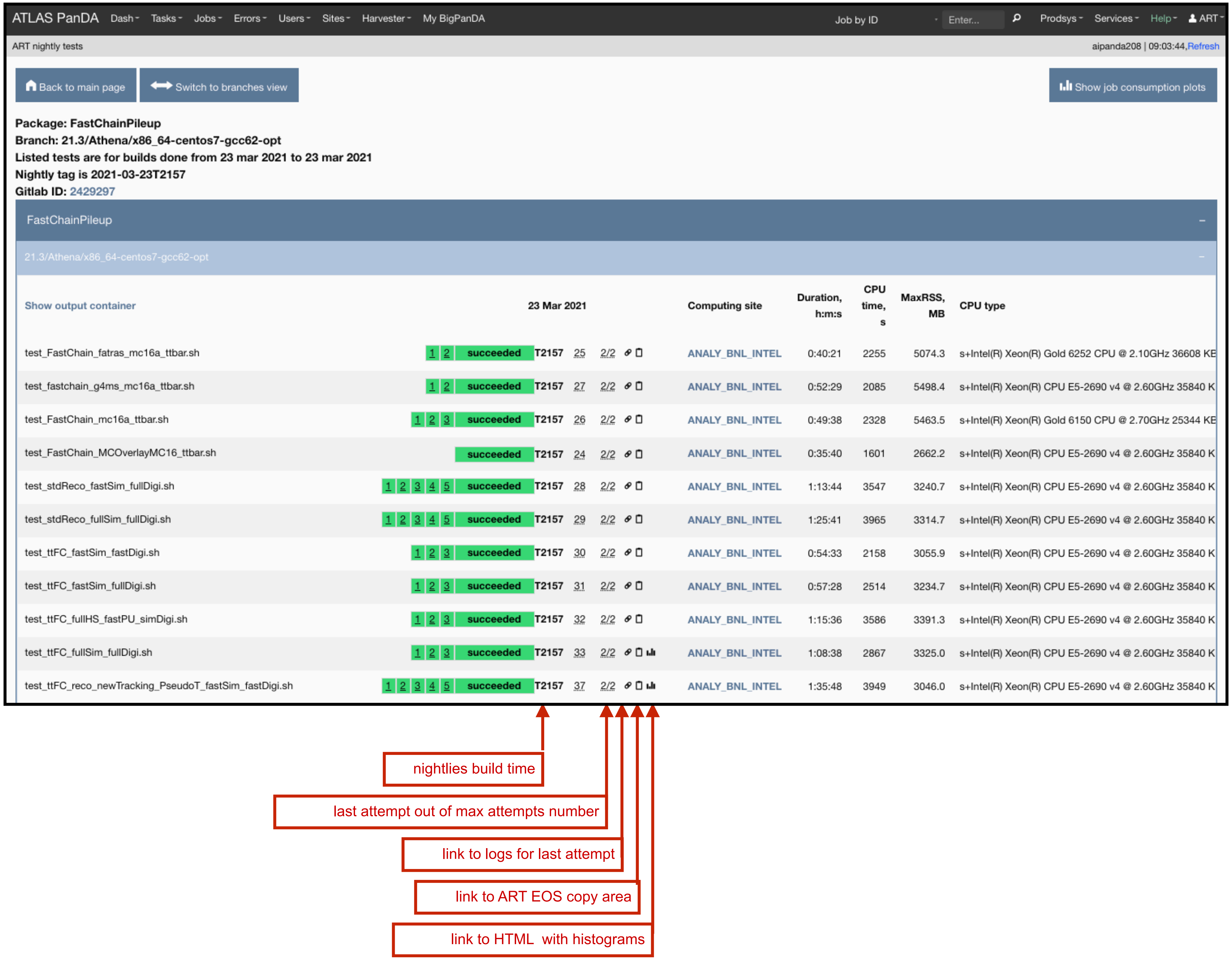
	19 Mar 2021	20 Mar 2021	21 Mar 2021	22 Mar 2021	23 Mar 2021	24 Mar 2021	25 Mar 2021
21.3/Athena/x86_64-centos7-gcc62-opt	0 23 0 0 T2156	---	---	---	0 23 0 0 T2157	---	---
master/Athena/x86_64-centos7-gcc8-opt	0 0 10 1 T2101	0 0 10 1 T2101	0 0 10 1 T2101	0 0 1 0 T2101	0 0 10 1 T2101	0 0 10 1 T2101	1 0 0 0 T2101

ART GRID jobs help

The color coding of test result is the following:

Active	- PanDA job still running.
Succeeded	- PanDA job finished, sub-steps results are not loaded yet;
Succeeded	- Athena and sub-steps succeeded;
Finished	- Athena succeeded, sub-steps failed;
Failed	- Athena failed or PanDA job failed;

Checking the ART grid jobs



ART nightly tests

aipanda208 | 09:03:44, Refresh

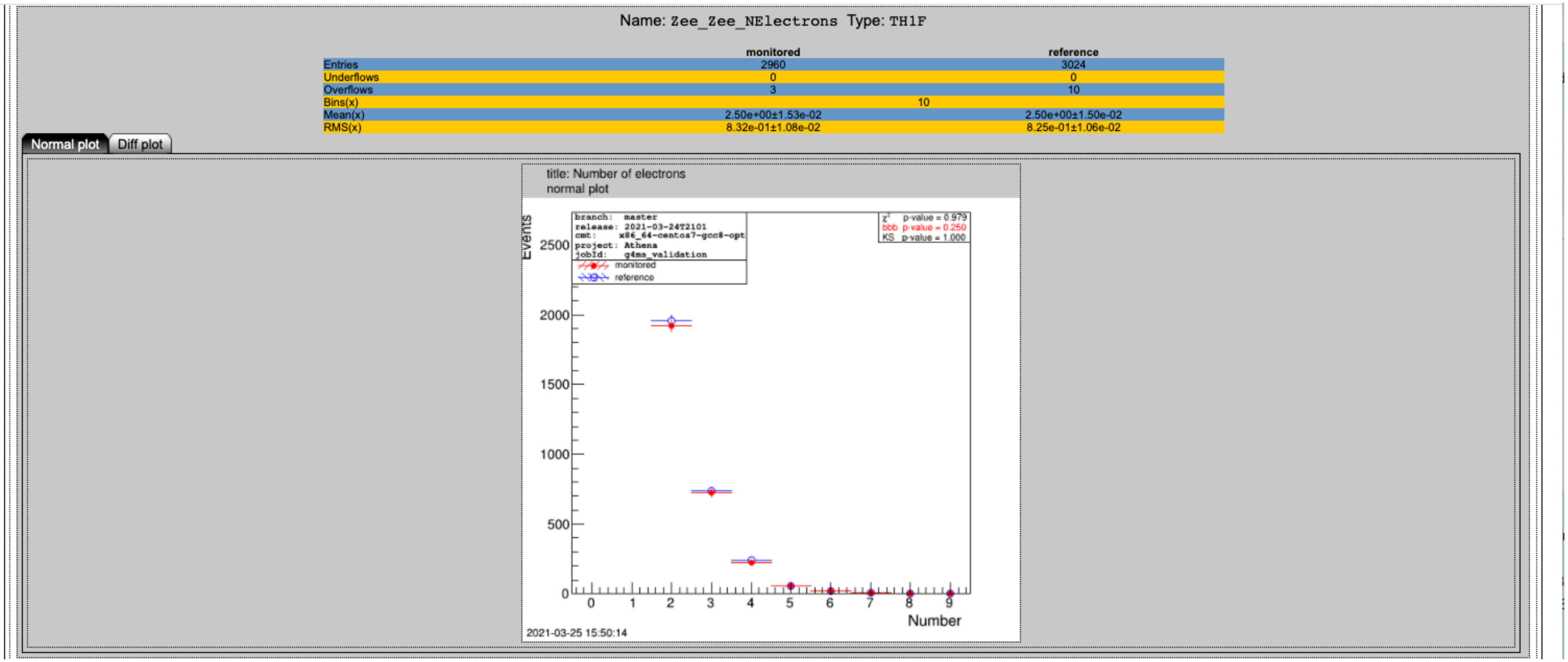
Package: FastChainPileup
Branch: 21.3/Athena/x86_64-centos7-gcc62-opt
Listed tests are for builds done from 23 mar 2021 to 23 mar 2021
Nightly tag is 2021-03-23T2157
Gitlab ID: 2429297

FastChainPileup

21.3/Athena/x86_64-centos7-gcc62-opt

Show output container	23 Mar 2021	Computing site	Duration, h:m:s	CPU time, s	MaxRSS, MB	CPU type
test_FastChain_fatras_mc16a_ttbar.sh	1 2 succeeded T2157 25 2/2	ANALY_BNL_INTEL	0:40:21	2255	5074.3	s+Intel(R) Xeon(R) Gold 6252 CPU @ 2.10GHz 36608 KE
test_fastchain_g4ms_mc16a_ttbar.sh	1 2 succeeded T2157 27 2/2	ANALY_BNL_INTEL	0:52:29	2085	5498.4	s+Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz 35840 K
test_FastChain_mc16a_ttbar.sh	1 2 3 succeeded T2157 26 2/2	ANALY_BNL_INTEL	0:49:38	2328	5463.5	s+Intel(R) Xeon(R) Gold 6150 CPU @ 2.70GHz 25344 KE
test_FastChain_MCOverlayMC16_ttbar.sh	succeeded T2157 24 2/2	ANALY_BNL_INTEL	0:35:40	1601	2662.2	s+Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz 35840 K
test_stdReco_fastSim_fullDigi.sh	1 2 3 4 5 succeeded T2157 28 2/2	ANALY_BNL_INTEL	1:13:44	3547	3240.7	s+Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz 35840 K
test_stdReco_fullSim_fullDigi.sh	1 2 3 4 5 succeeded T2157 29 2/2	ANALY_BNL_INTEL	1:25:41	3965	3314.7	s+Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz 35840 K
test_ttFC_fastSim_fastDigi.sh	1 2 3 succeeded T2157 30 2/2	ANALY_BNL_INTEL	0:54:33	2158	3055.9	s+Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz 35840 K
test_ttFC_fastSim_fullDigi.sh	1 2 3 succeeded T2157 31 2/2	ANALY_BNL_INTEL	0:57:28	2514	3234.7	s+Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz 35840 K
test_ttFC_fullHS_fastPU_simDigi.sh	1 2 3 succeeded T2157 32 2/2	ANALY_BNL_INTEL	1:15:36	3586	3391.3	s+Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz 35840 K
test_ttFC_fullSim_fullDigi.sh	1 2 3 succeeded T2157 33 2/2	ANALY_BNL_INTEL	1:08:38	2867	3325.0	s+Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz 35840 K
test_ttFC_reco_newTracking_PseudoT_fastSim_fastDigi.sh	1 2 3 4 5 succeeded T2157 37 2/2	ANALY_BNL_INTEL	1:35:48	3949	3046.0	s+Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz 35840 K

Histogram comparison



Name: `Zee_Zee_NElectrons` Type: `TH1F`

	monitored	reference
Entries	2960	3024
Underflows	0	0
Overflows	3	10
Bins(x)	10	
Mean(x)	2.50e+00±1.53e-02	2.50e+00±1.50e-02
RMS(x)	8.32e-01±1.08e-02	8.25e-01±1.06e-02

title: Number of electrons
normal plot

```
branch:   master
release:  2021-03-24T2101
cmt:      x86_64-centos7-gcc8-opt
project:  Athena
jobId:    g4ms_validation
```

χ^2 p-value = 0.979
bbb p-value = 0.250
KS p-value = 1.000

2021-03-25 15:50:14

Reference files are distributed on the grid

Used Technologies

docopt.py: To handle the command-line and its options

yaml and **json**: For configuration and status files

gitlab-ci: To submit nightly tests and wait for their results

open stack Virtual Machines (VM): To run all the gitlab-ci jobs on (installed/registered centrally via puppet)

docker and docker-images: To have the same environment on all the VMs

BigPANDA: For GRID job submission and monitoring

Rucio: To download results into the VMs

EOS and xrdcp: To copy results back from the VMs into EOS

asciidoc and **asciidocter**:

- To write the ART Manual
- To convert the asciidoc manual to pdf and a website

Summary

- ART is a framework to test the ATLAS offline software
- ART is fully in production and replaced a system that was bound to legacy infrastructure
- ART continues to evolve depending on the needs of ATLAS and on the evolution of the underlying infrastructure