Optimization of the merging procedure

Marian Ivanov, Cristian Ivan

Motivation (0)

One of the biggest problem for the (C)pass 0 and (C)PassX calibration is the merging

For many runs form the year 2010 and 2011 the manual intervention needed

- Download files locally
- Do merging
- Extract OCDB
- Update Savannah bug report – OCDB modification request

TPC alignment was not yet integrated in the default Pass calibration because of the memory consideration during the merging

Motivation (1)

The new TPC dEdx algorithm (~2.4 times lower gain) the run by run calibration of the correction necessary

- Merged trees as a input data
- Impossible to use standard merging anymore

Goal: Write robust merging procedure as a general jdl job

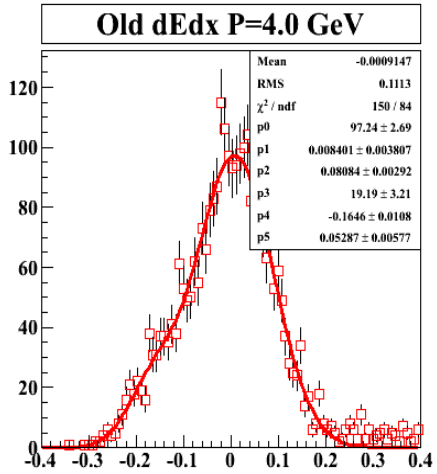
- Focus on the physics/calibration

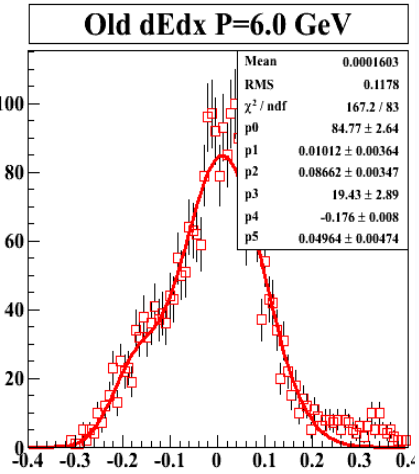
Time scale : 1-2 weeks

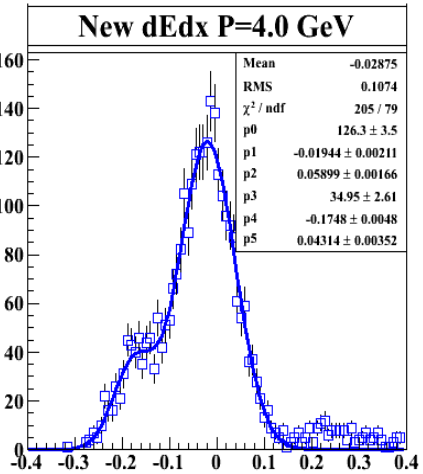
- Blocker for the reconstruction of the 2011 data

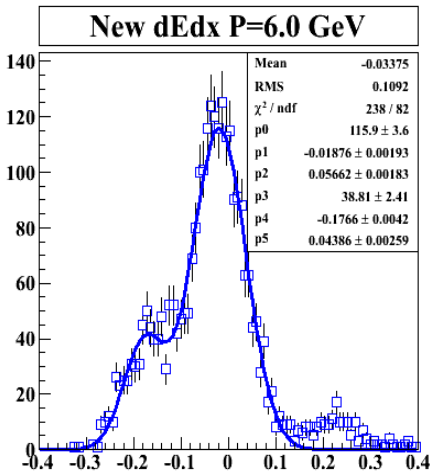
Comment: Cpass0, Cpass1 will start with the default merging

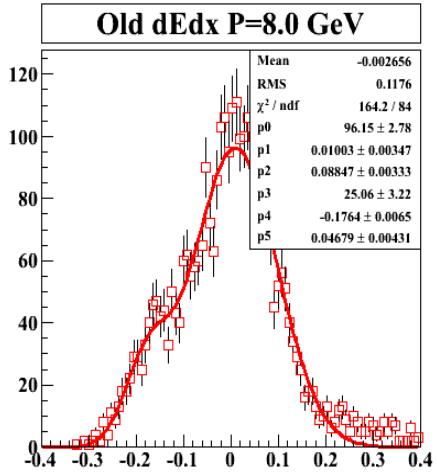
Dedx Performance for the -70 V setting – 2011 data

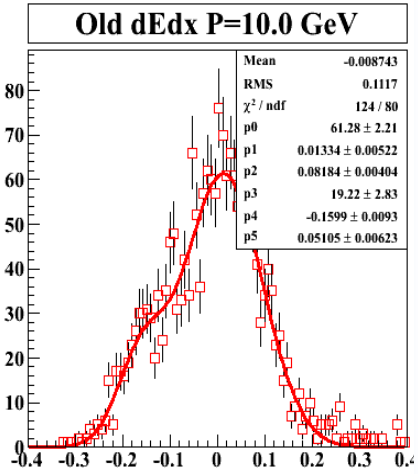


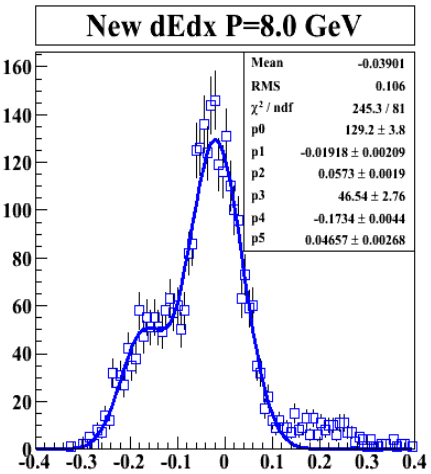


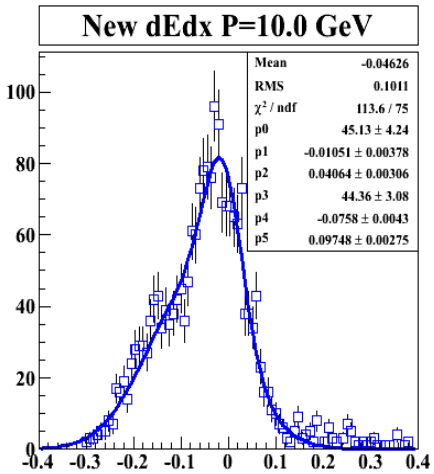












Merging problems

Availability of the input data to merge

- How fast we can access the data localized on different storage elements
- Merging time usually much smaller than access time

Memory consumption during the merging

- Limits on the memory consumption per process
- Slowness of the merging once started to use the swap space

Both problems are solvable by the change of the merging procedure

Alien access optimization

Consideration - Availability of the input data on the grid

Consideration:

- Merging jobs are not typical Grid/AliEn jobs – jobs can not be sent where the data are, as the input data for merging are distributed over several storage elements.
  - It is not typical job
- The availability of the storage elements is not 100 % guaranteed. Situation is changing dynamically.
- The speed of the download is tier to tier dependent.

Optimization of the data access on the Grid for merging

The input data are usually mirrored on several Tiers centers

The speed of the merging can be optimized tuning the time out parameters of the xrootd access

The total time is determined by the outliers in the behavior

Internal implementation in root classes:

- TAlienFile – derives from the TXNetFile
- Internally TalienFile query the file catalogue and provides the list of all the xrootd servers where the data are
- TxnetFile process the data in order, as provided by the TalienFile interface
- XRD default timeouts are used during the further data processing
- “Dead SE” are usually not excluded from the list of available SE

TXNet – xrootd timeout tuning – Magic lines

```
TGrid::Connect("alien");
//
gEnv->SetValue("XNet.RequestTimeout", opentimeOut);
gEnv->SetValue("XNet.ConnectTimeout", opentimeOut);
TFile::SetOpenTimeout(opentimeOut);
//
gEnv->SetValue("XNet.TransactionTimeout",downloadTimeOut);
```

To eliminate dead/not responding SE =>

- Modify the time-out for the file opening

The bandwidth between different storage elements varies by orders of magnitude

- Modify the time-out for transaction

TXNet – xrootd timeout - Test

```
TGrid::Connect("alien");
//
gEnv->SetValue("XNet.RequestTimeout", opentimeOut);
gEnv->SetValue("XNet.ConnectTimeout", opentimeOut);
TFile::SetOpenTimeout(opentimeOut);
//
gEnv->SetValue("XNet.TransactionTimeout",downloadTimeOut);

,,
```

Test Input data – PWG1 QA train
alien:///alice/data/2011/LHC11c/000153232/ESDs/pass1/QA66/

- 998 files to merge – mirrored on 4 SE

Parameters of test:

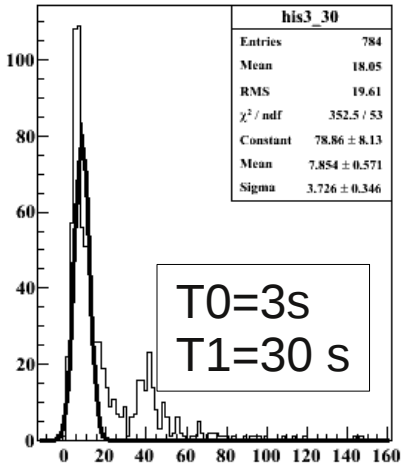
- Open timeout – 3, 5 s
- Transaction timeout – 30 s, 60 s
- Default timeouts

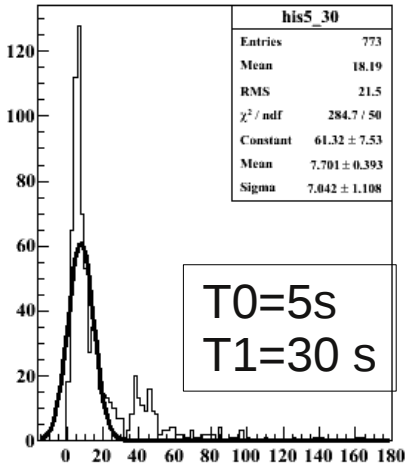
Parameters to monitor:

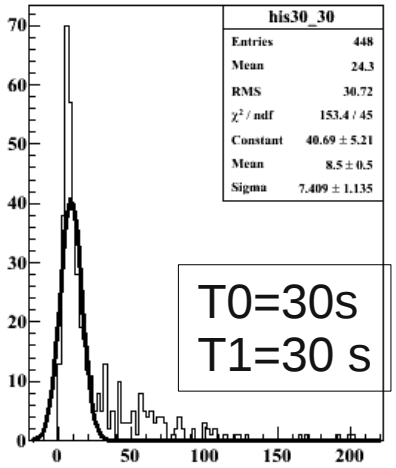
- Mean time per file
- Success rate
- Optionally – collect the “black list” (not part of the presentation)

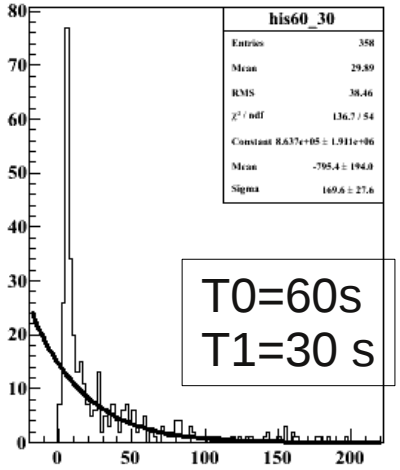
Results of the default tests are not reproducable - 2 Storage elements were down during the test.
Input data localized also on T2.

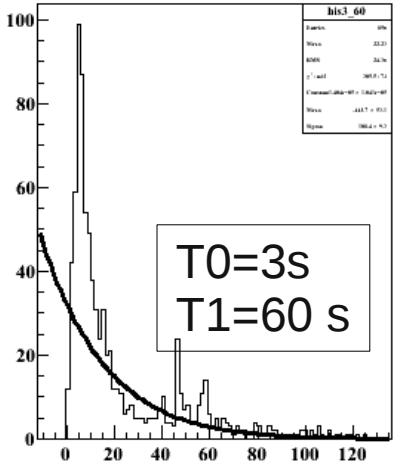
Result of time-out tests

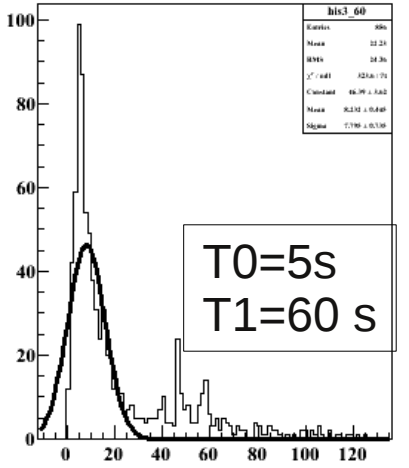


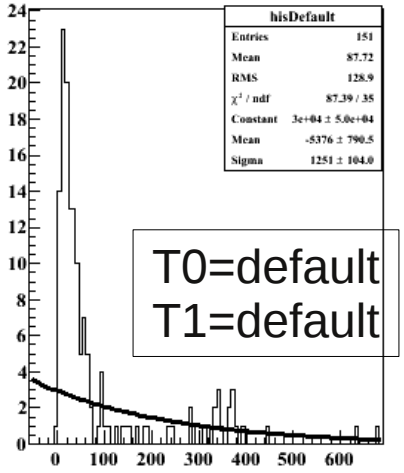




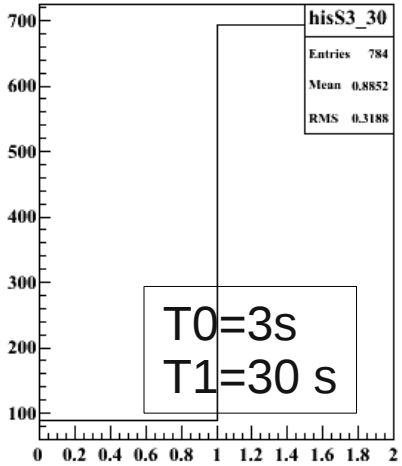


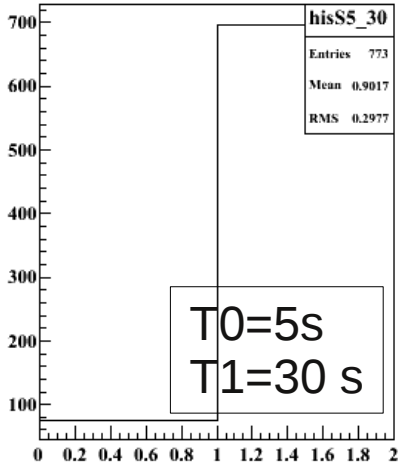


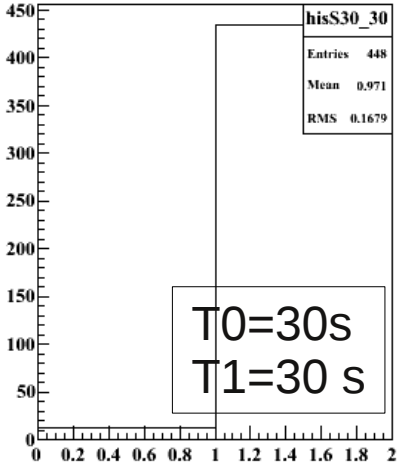


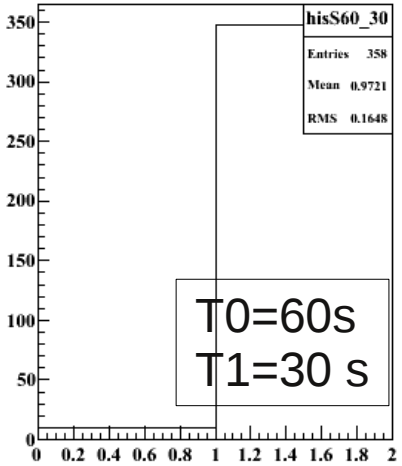


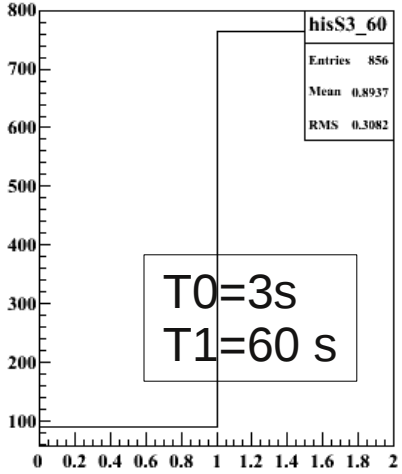
Success rate - Result of time-out tests

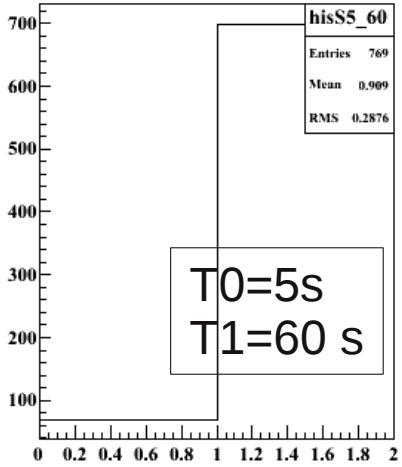


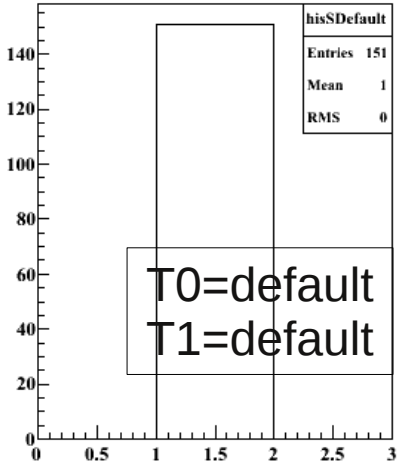












MI time-out Conclusion

The mean time determined by the outliers

- Mean to truncated mean ~ factor 10 for default setting

The time out for the file opening is essential

- If you can accept some loss of efficiency
- Fast solution for current merging ?

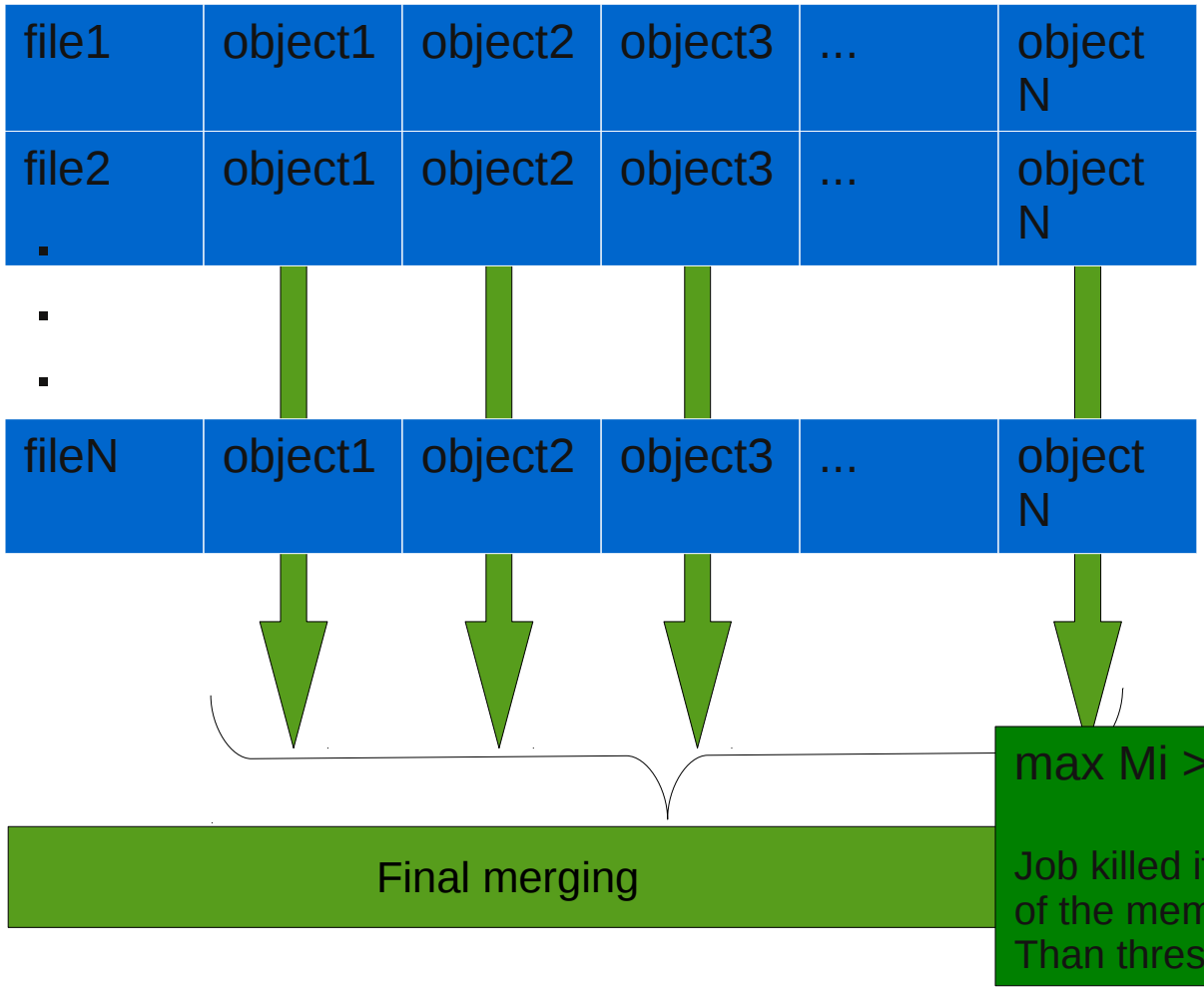
The mean access time with timeout ~ 10 s is about 10 time bigger than typical merging time

The mean access time can be significantly improved, running few caching processes (to local disk) in parallel.

- Optimally the caching time of the same order of magnitude as an merging time

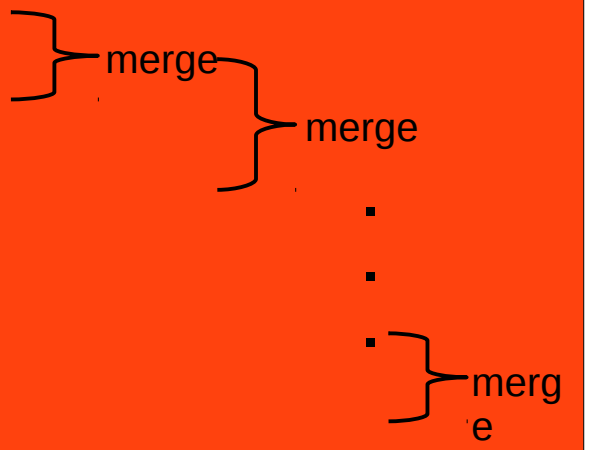
Memory consideration

Merging root files



Default merging

Merge file by file



max Mi > Mc

Job killed if max object of the memory bigger Than threshold

ΣMi > Mc

Job killed if the sum of the memory bigger Than threshold

Default merging: the whole chain can fail if a single file is not accessible and also uses more memory

Proposed sollution: if we copy files locally **with a time-out** and merge sequentially **object-by-object** we avoid alien instabilities and use less memory