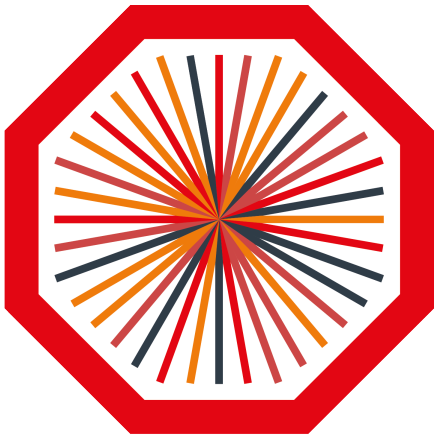
ALICE



Lessons Learned & What We Would Have Differently

Costin.Grigoras@cern.ch

Outline

- Object model
- Data model
- Monitoring

Object model

- Complex objects
  - Deep object hierarchy
  - Many branches (~800/ESD, ~400/AOD)
- Good for flexibility
  - Any production, central or user analysis runs on one of the 2 formats
- Bad for I/O
  - Large event sizes
  - Depends on AliRoot objects
  - Large deserialization penalty

Flat AODs

- Similar number of branches
- Better compression, smaller overhead
- Doesn't solve the I/O problems
- Needs a lot of effort to develop and maintain

NanoAODs

- Analysis-specific objects
  - Filtered once, reused many times
- Might solve the I/O problem
  - Flat, as small as needed, structures
- But not necessarily feasible
  - Analysis touch most of the current branches
  - Interferes with the train model
- Also harder to manage centrally
  - All central productions are nicely accounted for

Current data model

- Jobs go to where a copy of the input data is
- Uploading 2 (or even 3) copies of the results to the nearby SEs
- Single catalogue for all files
- Single protocol (Xrootd) to access data
- Central authorization service

And problems

- Not enough I/O throughput for the analysis
  - Didn't foresee a throughput/hepspec requirement
  - The hepspec to local storage volume one was very inspired
- Wishing there wasn't a CPU / Storage split
  - Hadoop model
- Too many layers between data and processing
  - And too many flavors of them
  - Loosely coupled is good … or not ?

Keeping me busy

- “We have to remove some out-of-warranty xrootd servers and replace them by new servers. Apart from dealing with the data that we will have to discuss later, some choices have to be made”...
- “Several ALICE files (*) on our HPSS instance were lost last week. “
- “We have to evacuate also data from one of our disk server : aa.bb.cc. The size is about 50To. But we don't have now free space.”
- “We are very sorry to have to inform you that we lost the storage filesystem on“...

Consequently

- To sync the content of catalogue and SEs requires patience

```
Removed 19206246 files (383.2 TB), kept 91392933 files (3.789
PB) from ALICE::CERN::EOS, took 38d 16:08
… ALICE::CNAF::SE, took 31d 16:59
… ALICE::PRAGUE::SE, took 57d 2:48
```

  - Because `*xrd rm ...*` exited with exit code 0 even if there was an error and I wasn't checking further
- Procedure to continuously check input data of suspiciously failing jobs
  - Replicas gone, without notice
  - Corrupted content, requires full download of all replicas
  - Inaccessible data servers

Monitoring

- High-level views critical to understand the system
- In-detail monitoring useful for debugging the components
- Most important is how to use this data to take automatic decisions
  - Quite happy with the SE discovery mechanism we have

Missing data

- Many blind spots, often end up in long debugging threads with the admins
- SE health status, having to trust the site to handle it
- No self-healing mechanism
  - EOS is a good example of lessons learned
- The opposite is also true, it is trivial to generate more monitoring than experiment data

Also on the wish list

- Updates push mechanism
  - Almost all 5yo+ Xrootd versions have been updated, moving on to the 3.1.0 and higher :)
- Time to upgrade the Xrootd client in AliEn

Summary

- The good
  - Single catalogue for all files, single protocol to access them
  - Job brokering to data
  - Automatic data placement
- The bad
  - I/O is a bottleneck, for some activities at least
  - Separated CPU from Storage is part of the problem
  - Monitoring was an afterthought
- The ugly
  - Manual interventions on broken components
  - Many layers between the processing code and the data