CMS

outline

- Current CMS computing model (with focus on specificities)
- Some projections to medium/long term future
- Some ideas

What is our data?

- **RAW**: dump of CMS FED buffers
- **GEN**: the output of Physics Generators (think of it as a LHE format in ROOT)
- **SIM**: Geant4 Hits
- **DIGI**: the simulated CMS electronics response to a given event
- **RECO(SIM)**: the output of reconstruction, with most data products retained
- **AOD(SIM)**: a slimmed version of the previous. Detailed Hit by Hit local reconstruction dropped. It is the RunI preferred analysis format
- **MINIAOD(SIM)**: a physics oriented slimmed format. Designed to cover 90%+ analyses in RunII
- **NANOAOD(SIM)**: a ntuple based compressed format; under validation, should cover 50%+ of the RunII analyses
- **Combined tiers** (GEN-SIM-DIGI-RAW for example) possible

- Typical Sizes per event (RunII PUs)

Data Tier	Size (kB)
RAW	1000
GEN	< 50
SIM	1000
DIGI	3000
RECO(SIM)	3000
AOD(SIM)	400 (8x reduction)
MINIAOD(SIM)	50 (8x reduction)
NANOAOD(SIM)	1 (50x reduction)

Analysis data formats

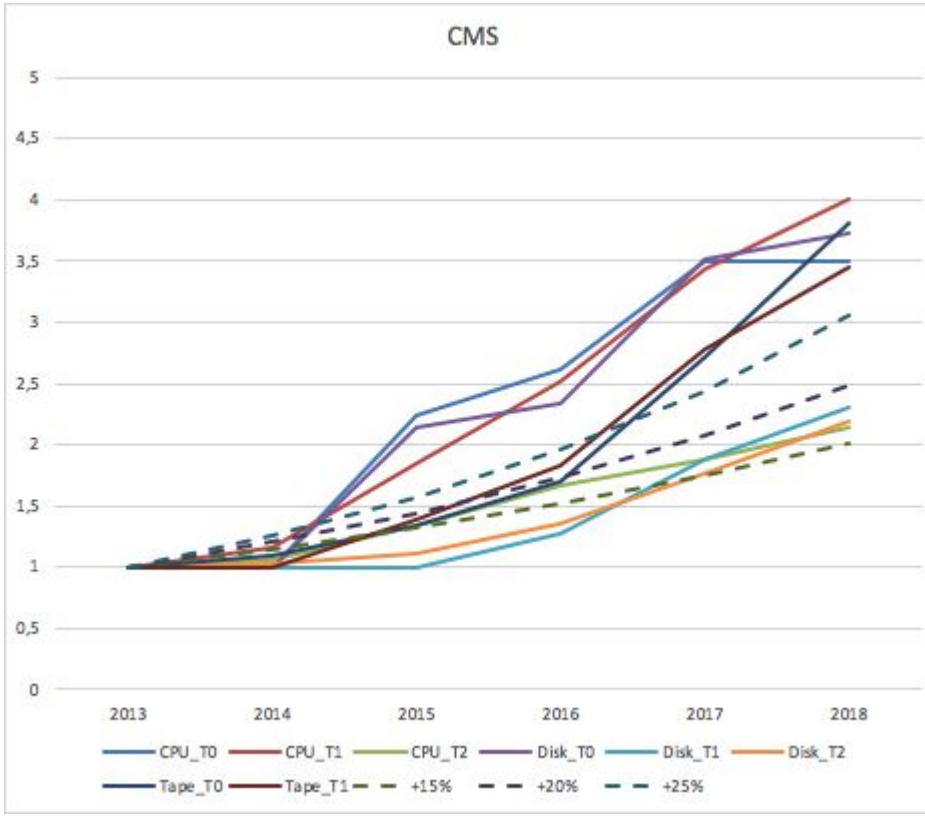
Exec Summary: "Prevalent analysis format in CMS reduced by a factor 3000x in event size since the start of RunI

CMS Computing Model (data oriented view)

- 2018:
  - ~150 PB of Disk + ~260 PB of tape pledges resources
  - Distributed in 7 Tier-1s and ~50 Tier-2s
-

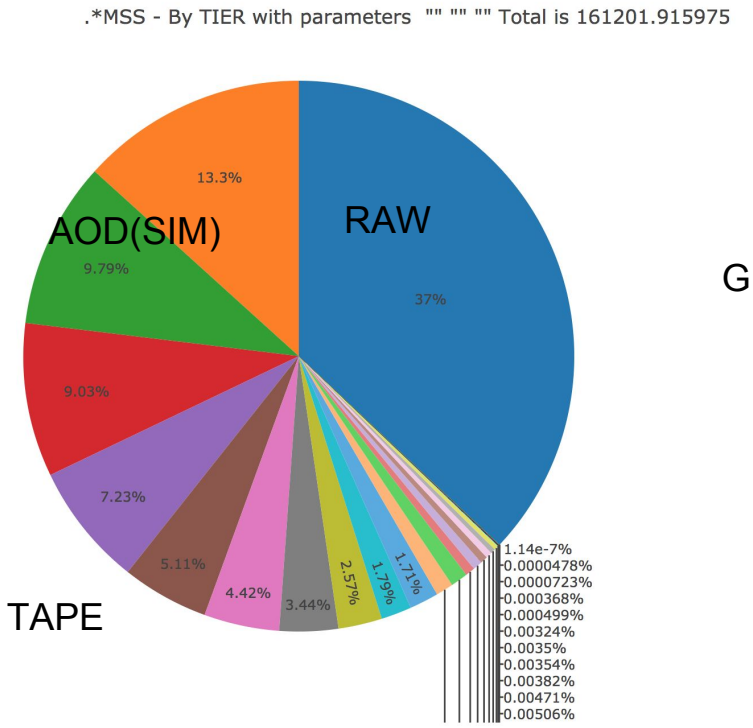
Tier	Pledge Type	CMS	Required	Balance
Tier 0	CPU (HEP-SPEC06)	423,000	423,000	0%
Tier 0	Disk (Tbytes)	26,100	26,000	0%
Tier 0	Tape (Tbytes)	97,000	99,000	-2%
Tier 1	CPU (HEP-SPEC06)	561,540	600,000	-6%
Tier 1	Disk (Tbytes)	55,338	60,000	-8%
Tier 1	Tape (Tbytes)	165,505	188,000	-12%
Tier 2	CPU (HEP-SPEC06)	929,732	900,000	3%
Tier 2	Disk (Tbytes)	65,691	70,000	-6%

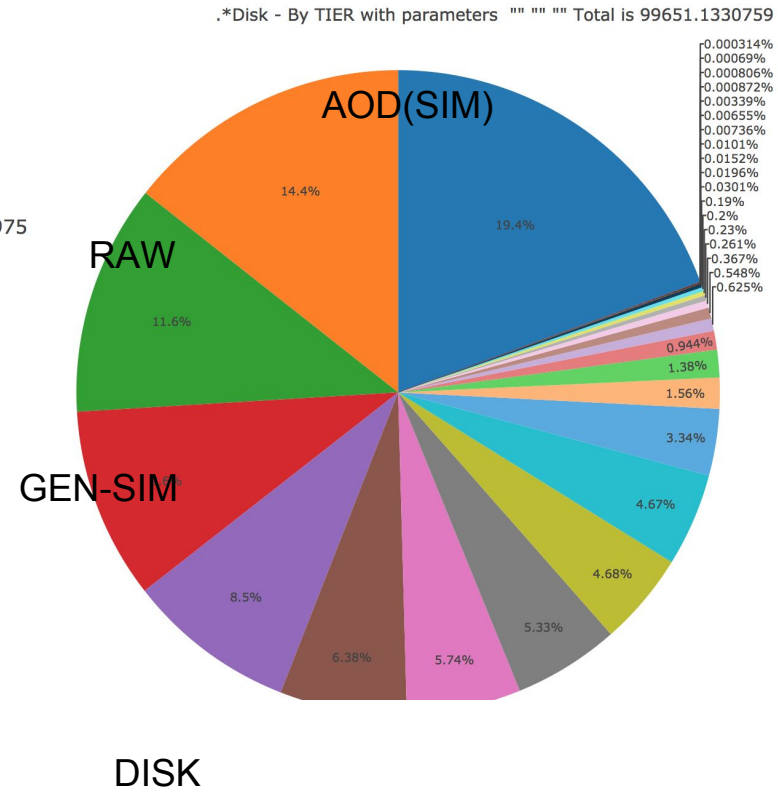
- Evolution since the end of RunI
  - Tape capacity increased by > 25%/y
  - Disk increase much milder (exception is T0)



A snapshot of what we have on disk/tape

(taken at one moment where we are quite unbalanced with PhaseII preparation samples and doing data and MC reprocessing)



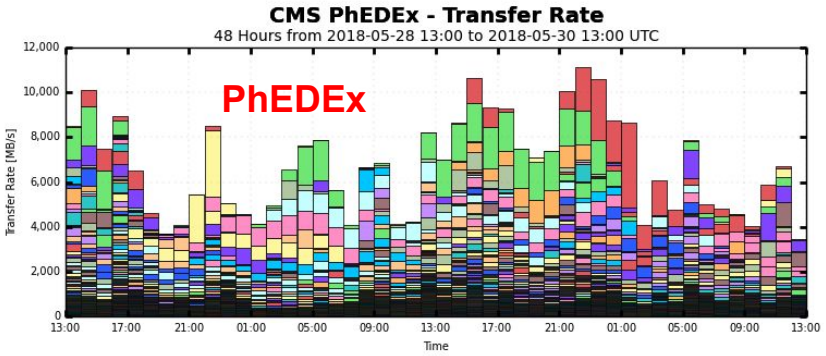


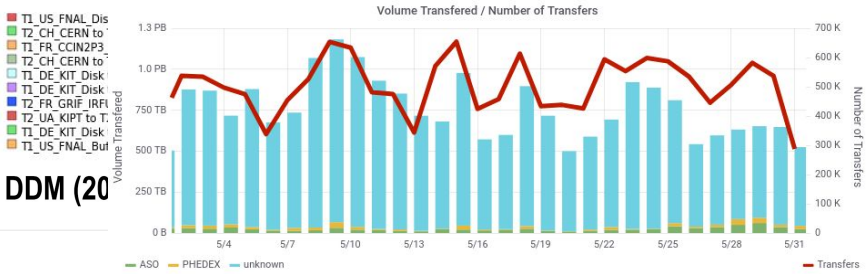
What do we have as DM?

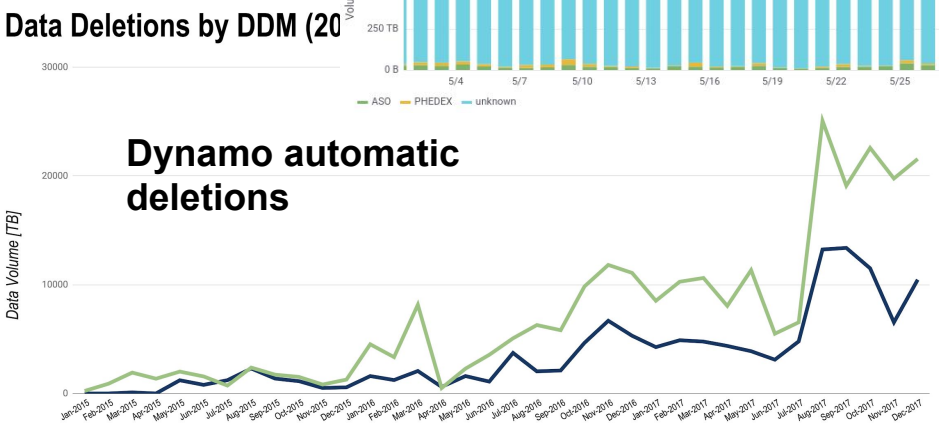
- Currently, 2+1 products: PhEDEx and Dynamo + ASO
- PhEDEx:
  - transfer work-horse. Uses FTS for most of the transfers. It as written pre-FTS: it can handle SRM/Xrootd/... directly
  - Takes care of routing (only Dataset destination is specified)
  - Handles tape stage in/out
  - Knows where files are in the system, and can perform consistency checks (with manual effort and help from sites)
  - Designed to have per site agents (eventually not sitting at site itself)
    - This was initially done in order to locally optimize systems (posix access, custom tape interactions)
- Dynamo:
  - Layer working on top of PhEDEx
  - Interacts with production systems in order to populate CMS storage areas
  - Based on popularity figures, decides which datasets to spread in more than the minimum # of copies, and evicts from disk rarely used ones
  - Able to perform automatic consistency checks
  - Policy driven
  - Also handles (some) T0 disk buffers
  - Rebalances sites and handles problematic sites
- ASO:
  - Different product, handles Analysis output transfer to user sites.
  - No other lifetime handling (no deletions, no further movement)

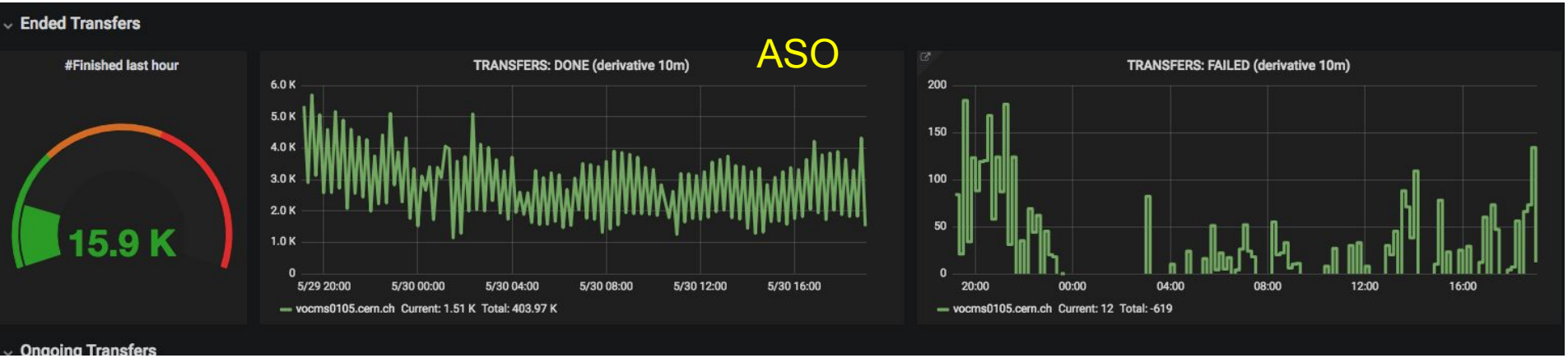
What happens “daily”?

- Again, a “random day” snapshot:
  - 600k files moved
  - 800 TB moved
  - Average files from prod ~ 3 GB
  - Average files from analysis ~ 100 MB
- This does **not** include all the xrootd remote file access









Planned transition of DM stack

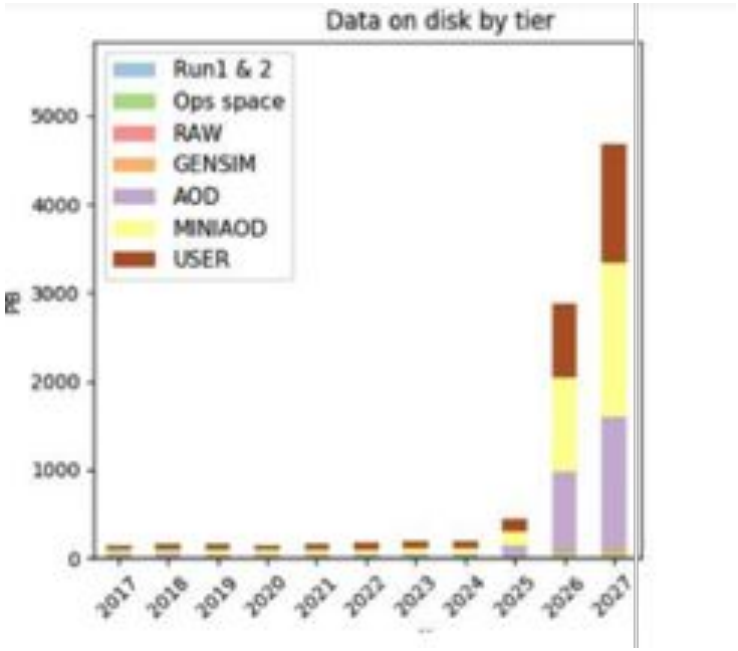
- DM review ongoing; scope is the identification of a solution
  - Which can scale to PhaseII
  - Which has a clear support model (Institutions inside/outside CMS)
  - Which is able to serve Analysis and Production files at the same time
  - Which is able to operate in a series of identified CMS use cases
    - Deployment as a CMS service: disaster recovery, replicability and relocatability on different systems
    - Dealing with problematic and temporary sites
    - Dealing with corrupt and missing files
    - Dealing with large deletion campaigns
    - Implementation of a lifetime model for datasets as recommended by ECoM17
    - Dealing with automatic deletions via policies
    - The capability to implement complex policies which include selections on at least: dataset names and tiers, sizes, sites, lifetime, production status, geographical location

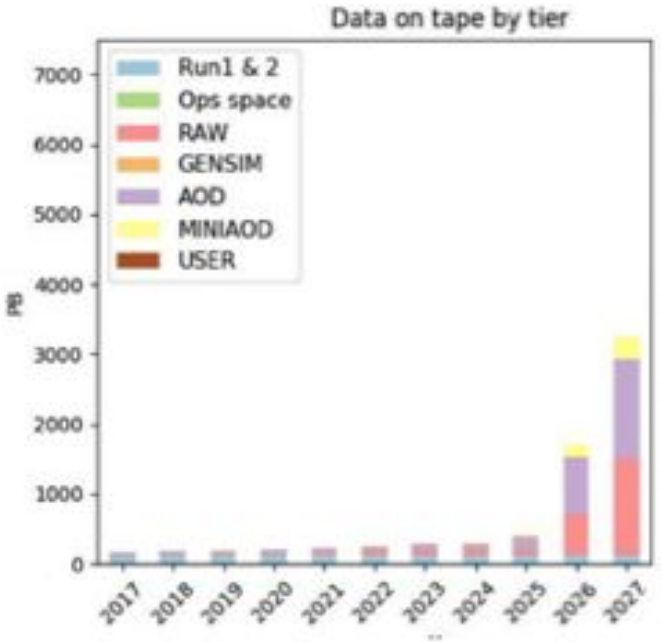
- Two solutions identified upfront:
  - An evolution of Dynamo, directly interacting with FTS
  - Rucio (ATLAS, now being used/tested by other experiments)

We expect a report by the Summer; management decision will follow shortly

Projections to RunIV

- We intentionally use last “official” LHCC figures (Sept 2017); we expect an yearly update at the Sept LHCC
- Rule of thumb is that by “not doing a lot” we would be off with respect to flat budget by 4-6x
- Explicitly not using NanoAOD in this modelling (we will try and elaborate on this @ next iteration)

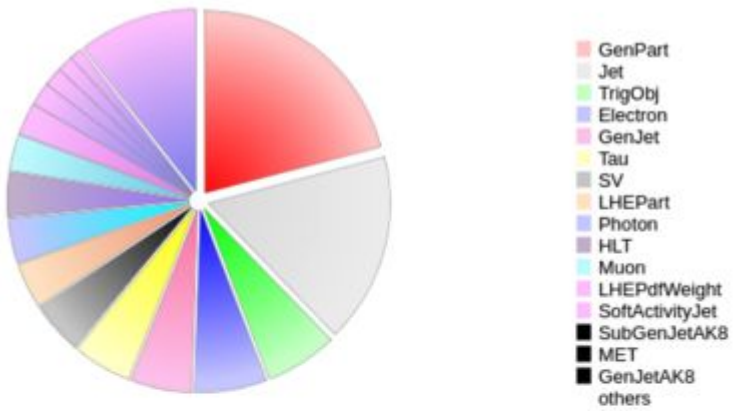




Some concepts

(from here on it is some is the opinion of the speaker, admittedly, ...)

- The WLCG strategy document, public since last week, contains some of the ideas we have been pushing as a solution to large volumes of storage
  - **NanoAOD**: 1 kB/ev @ RunII → 4-5 kB/ev @ RunIV. **If** we can make sure the analysis acceptance @ 50%+, removes a lot of pressure from storage (total, # of files, # of transfers, ...)
  - **Caching layers** on Demand and distributed caches
    - Solution to enhance access to data in well connected regions
    - Solution to better use opportunistic centers
    - Solution for disk-less centers
  - CMS is already now increasing over and over the **remote access** to its data



Event data

collection	kind	vars	items/evt	kb/evt
GenPart	collection	9	53.32	0.330
Jet	collection	31	8.69	0.266
TrigObj	collection	11	9.70	0.101
Electron	collection	48	1.15	0.100
GenJet	collection	7	7.70	0.085
Tau	collection	38	1.33	0.082
SV	collection	13	2.79	0.073
LHEPart	collection	6	7.00	0.063
Photon	collection	28	1.50	0.062
HLT	singleton	569	1.00	0.061
Muon	collection	33	0.76	0.050
LHEPdfWeight	vector	2	33.00	0.044
SoftActivityJet	collection	4	5.96	0.031
SubGenJetAK8	collection	5	2.24	0.026
MET	singleton	11	1.00	0.022
GenJetAK8	collection	7	1.15	0.016
FatJet	collection	20	0.31	0.016
LHEScaleWeight	vector	2	9.00	0.016
SubJet	collection	14	0.41	0.014

The data lake

- Indeed one of the most promising solutions
  - We should not rule out others yet - still space for other bottom up solutions
- Why interesting?
  - **Reduce** complexity as seen by the Experiments: fewer endpoints, leave some actual replication / routing / destination decisions to the lake itself, ...
  - **Reduce** total operation cost: automatic handling of failing sites / failed transfers, automatic enforcement of QoS policies, ...
  - **Reduce** expensive disk wrt to cheap disks / tapes via tiered storage
  - **Faster** response / bandwidth via internal / transparent scaleout (no strict DS to Site association)
  - **Better** tooling to incorporate non WLCG sites
  - **Built** from the start thinking also to non-WLCG sites
    - HPC, Opportunistic, Caches, SDNs, CDNs, ephemeral sites, ...

- Interested to see/work on the implementation

DataLake

- Details are not of scarce relevance (some of them are probably a longer term view than 2026 ....)
  - How many lakes?
    - If you ask me, not too many. Continental level at least?
  - How many supported storage “low level” solutions?
    - If you ask me, all those the major sites want to use
  - Should it include the provisioning of caching layers?
    - If you ask me, yes. As low as caches at infrastructure level
  - Should it include network provisioning?
    - If you ask me, yes. SDNs? How else?
  - Should it be seen from experiments as a single entity?
    - If you ask me, yes. Internal transfers, data replication, data locality, ... all via QoS policies
  - Should it be linked to Workload management systems?
    - If you ask me, not to specific ones. It should expose APIs to orchestrate with WM systems, though.

Conclusions

- Any idea / solution sketched here is clearly not able to do miracles
  - If you need to store N bytes you will need to buy storage for N bytes

- But we should aim at
  - Lower the support level needed (by the experiments themselves)
  - Offload mature components to the infrastructure
    - Parts of data management, caches, handling of network routes
  - Lower the complexity seen by the experiments
    - Fewer logical sites, QoS to be handled / enforced by the infrastructure