Analysis experience in CMS

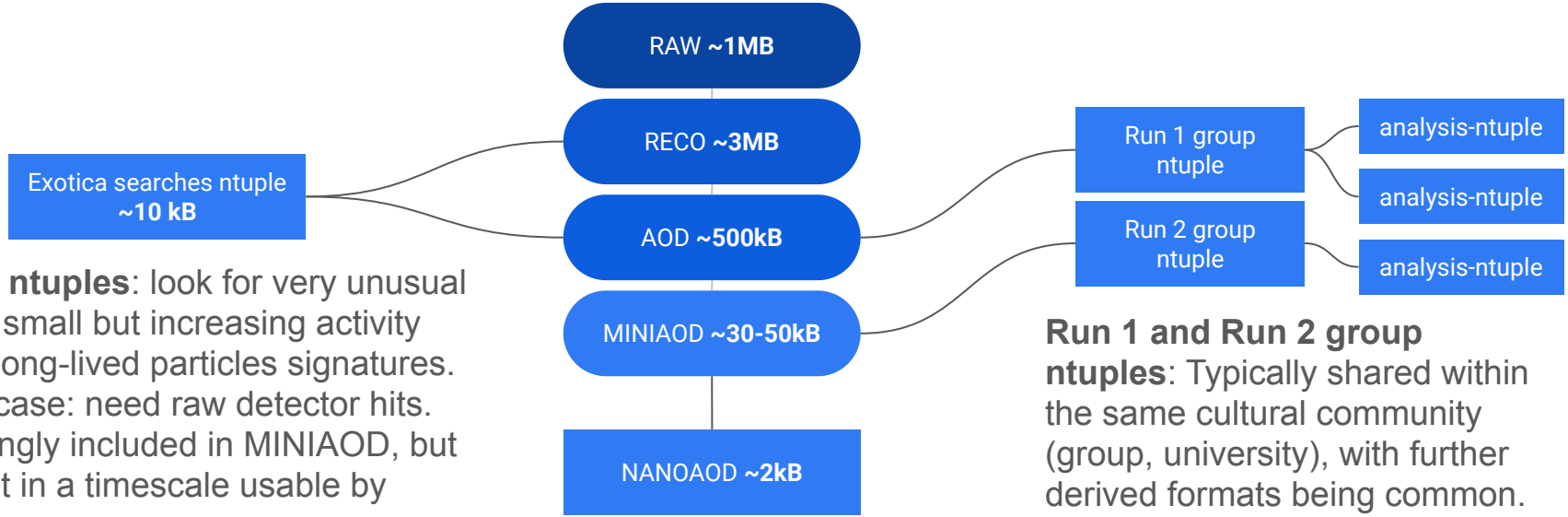
Joosep Pata (Caltech)

Disclaimer & thanks

- This is a subjective summary of a few analysis experience in Run 1-2
- Necessarily coarsened, cannot represent the full diversity of ideas, experiences, possibilities in a large collaboration
- Mostly subjective observations for grid, actual data for local analysis
    a. Extracting reliable and understandable data from grid monitoring needs expertise!
- Main take-away: **producing the ntuples on the grid is a bottleneck!**

Thanks for input to A. Rizzi, M. Peruzzi, L. Corcodilos, S. Cruz et al.

Formats overview



Exotica ntuples: look for very unusual signals, small but increasing activity around long-lived particles signatures. Typical case: need raw detector hits. Increasingly included in MINIAOD, but often not in a timescale usable by groups.

NanoAOD: shared between physics analysis groups (~50% adoption), increasingly used by object groups for routine tasks (e.g. calibrations).

Run 1 and Run 2 group ntuples: Typically shared within the same cultural community (group, university), with further derived formats being common.

1 kB/ev = 1 TB per billion events
1 MB/ev = 1 PB per billion events
Run 2 analysis ~ 5B events

Credit: M. Peruzzi, ACAT 2019

Typical workflow

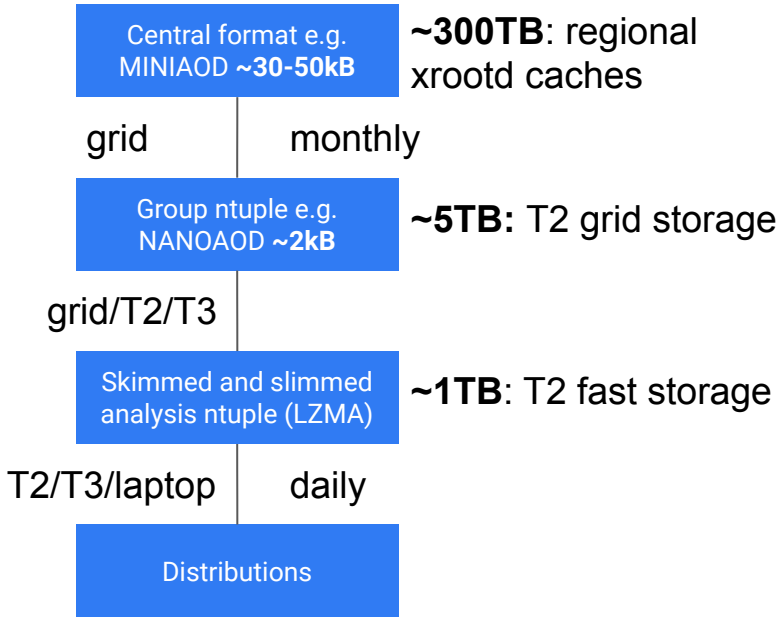
Our input is MINIAOD on the grid, distributed across sites or caches. For analysis, there are two main steps:

I. **Ntuple production**: From MINIAOD, group ntuples (e.g. NANOAOD) are produced that can be collected at a single site for analysis. Some analyses cannot use centrally-produced ntuples and must produce their own on the grid.
II. **Ntuple processing & analysis**: Ntuples are processed and analyzed on a daily basis to produce physics results. Sometimes, intermediate reduced formats are created from the ntuples.

Ntuples may get updated, which requires reprocessing (step I) and can be expensive.

Analysis overview: Higgs(μμ) with jets example

- Fully-featured Higgs analysis with systematics, jets, leptons, large background samples, involved corrections
- Producing the ntuples: **~10 Hz/thread, 300+ TB total read, days-weeks with ~10k threads on grid**
  a. Central production preferred, private production in case of need (happens often!)
- Processing the ntuples: hourly-daily basis
  a. **~1 kHz/thread, 1TB total read, few hours with ~500 threads on T2/T3** (may soon fit in one to a few dense servers)
  b. Testing can be done on a few files or very small subsets in real time
  c. Full dataset iteration is needed on a daily basis



Higgs(mumu) analysis IO and compute

- Infrastructure: isolated fraction of T2_US_Caltech
  a. ~500 CPU threads, CPU E5-2670 @ 2.60GHz with hyperthreading
  b. 100TB CEPH array of 35 enterprise HDD-s on 3 servers, ~2.5 GB/s sustained read
- Full systematics: ~1 kHz / thread, ~500 parallel threads, ~1 TB input
  a. Turn-around time about 2-3h, very predictable runtime and failures (0%)
  b. Typical job read speed 1-2 MB/s
  c. CPU-bound (number of local batch slots)
  d. In practice, full systematics preferred at a late stage of the analysis (always need the full result)
- Partial systematics: ~50 kHz / thread
  a. Turn-around time of ~20 minutes for the full dataset
  b. IO-bound at 2.5 GB/s
  c. Useful for quick checks and early in the analysis

Typical daily analysis tasks

- Apply a new correction to some datasets (partial reanalysis)
  a. If correction can be computed from existing quantities, need to just reanalyze some ntuples (fast)
  b. For consistency and bookkeeping, often prefer to reanalyze everything
  c. If new quantities are needed, need to additionally reprocess the ntuples (MINI → NANO, slow)
- Synchronize event or object selection with another group (partial)
  a. Understand why an event or an object in the event was dropped, useful to have access to some “unfiltered” samples
- Develop a new analysis method (partial)
  a. For some specific datasets, need to iterate and develop code on a real-time basis
  b. Typically done on just one file on an interactive node or laptop
- Debug odd cases in reconstruction (partial or full)
  a. Bugs may affect just a small fraction of rare events, need to process large samples to detect issues
- Update results with new calibrations (full)
  a. Reanalyze all ntuples with an updated calibration

Typical daily analysis tasks (cont)

- Retrain an MVA (partial reanalysis)
- Re-compute a few key variables, e.g. DNN or other CPU-intensive quantity (full)
    a. In extreme cases, this can get very expensive and require the grid (e.g. Matrix Element Method)
- Update systematics and recompute all derived quantities, CPU bound (full)
- Add a new dataset to the analysis, fetch it from grid and produce+analyze
- Add a new quantity to the ntuple (full production)
    a. A common but expensive operation, requiring interfacing with the grid and wide-area processing
    b. Highly-dependent on grid health and configuration
    c. Typical case: overflow to xrootd-access can decrease job efficiency & success rate significantly
        i. In practice, analyses try not to rely on xrootd streaming as much as possible (not ideal!)
        ii. This is partially mitigated by regional xrootd-caches with full maintenance, but still R&D
    d. Typical case: temporary failures of central database services (e.g. DBS/DAS) cause production to be lost
    e. If a few % of a dataset are lost in production, usually the best course of action is to reprocess all!

Common themes in Run 2

- Analyses are migrating to central ntuple formats (e.g. NanoAOD), shared between groups, joint production
    a. Extended central ntuple formats (>>2kB/ev) for specific detector-level calibration tasks (don't need all samples)
- The **bottleneck is generally in producing the ntuples**: Hz-level speeds, hundreds of TB of data to access over grid
    a. Batch/grid queues can be overloaded with CPU-inefficient jobs (xrootd transfers!)
    b. Wide-area data access in compute jobs perceived to be opaque and unreliable
    c. Small but rising trend in exotica: need to access low-level data for many samples to revisit reconstruction, prefer prestaging data locally
- Shared ntuples are copied locally whenever possible (~5-10TB total in Run 2)
    a. For this, sites need to have fast, easy-to-use scratch space, e.g. 100TB CEPH instance
- Analysis typically on local resources: ~ 1 kHz/thread , ~1 TB of data
- Minor changes to the analysis ntuple are an expensive operation

10x extrapolations based on Higgs(μμ) with jets

- Input to ntuple production: 500TB of MiniAOD → **5PB of MiniAOD**
- Production: 10k grid jobs for a week → **100k grid jobs for a week**
  a. Rely on central production and smaller private runs on grid
- Finished ntuples: 5TB of NanoAOD, 90% shared between analyses → **50TB**
- Skim-slim: 1TB of analysis-specific relevant data → **10TB**
- Processing and analysis ntuples (1-2x per day)
  a. Full systematics: 500 threads @ 1 kHz → **5000 threads or ~200 GPUs** (currently 15x improvement seen)
  b. Partial systematics require x50 less computing
  c. Aim to run overnight on T2/T3
- Daily development on 1-10% of the dataset on a desktop is possible (see typical analysis tasks)

Code Quality

- Comments and documentation are useful but they need to be written :)
- Short ad-hoc test jobs, consistency checks before large runs are common
- Unit tests for some “library” functionality is possible, but generally not driving the analysis code development due to experimental nature
- Automatic tests (CI) run on various small samples to verify no unexpected change in distributions has been useful
- Large-scale continuous running of analyses needs infrastructure, even ~100 threads at CERN gitlab with local data can be a great resource!
- Large performance increases from simple CPU profiling in analysis code
  - CPU is not usually the limiting factor so might not care
  - E.g. compute jobs with xrootd reading might spend a significant amount of time waiting for data
- Mostly foresee to use the same code or similar in Run 3, extended use of RDataFrame

Summary

- Primary tasks: **production** and **processing** of ntuples
- Analyses are primarily bottlenecked by **production of ntuples from MINIAOD**
  a. Need to run on grid with a large scale but many points of failure and often opaque response
  b. Network, storage, CPU bottlenecks
  c. Job inefficiencies and failures due to a very diverse environment
- Growing class of exotic analyses that must access low-level data
  a. Benefit from temporary partial staging of data
- Processing of ntuples with daily turn-around time is feasible on a T2/T3
  a. Given local storage (e.g. 100TB Ceph covers all of Run 2 for Caltech)
  b. CPU-efficient codes (e.g. RDataFrame, coffea, python+numba)
  c. Use of GPUs for DNN evaluation
- Developing on a fraction of data in “real time” on personal computing resources could be feasible

Backup

Data formats

1. **Lucas et al: b*-> tW all-hadronic analysis in B2G**
    a. NanoAOD + NanoAOD-tools, mainly for JME modules
    b. Data accessed both remotely and locally, xrootd EOS->local is a bottleneck
2. **Sergio et al: SUSY 2OSSFL, ttH multilepton**
    a. NanoAOD + NanoAOD-tools
    b. Skim MC to about 10%, analysis-specific ntuples
3. **Joosep et al: HiggsMuMu, ttH(bb)**
    a. Official unprocessed NanoAOD, about 5TB for full Run 2 data + MC
    b. optional HLT bit selection, branch dropping & recompression, 1B events @ 1TB
    c. no intermediate ntuple format, keep the same branch structure
4. **Si et al: Razor/LLP: SUSY, VVV, EXO, ...**
    a. Based on RAW/RECO/AOD, ~2-10 kB/event
    b. May require very low-level information used by several groups
    c. Cannot fix to a schedule due to on-the-fly nature of EXO: “the only constant is change”

Processing

1. **Lucas et al**
    a. 10k jobs to remake all distributions, with the longest job taking ~2 hours
    b. limited by NanoAOD-tools JME modules
    c. With RDataFrame, less resources needed, but need to port everything
2. **Sergio et al**
    a. Strict skimming (ttH multilepton) or slimming (SUSY)
    b. Processing speeds around a few hundred Hz
    c. Local-IO limited or CPU limited (in case of many MVAs)
3. **Joosep et al**
    a. Full systematic analysis code with CPU/GPU acceleration, directly to histograms
    b. Process full Run2 data+MC up (~1B events, ~1TB, ~1 kHz/thread), up to few times a day on ~700 threads
    c. Bottlenecks:
        i. Adding minor branches to NanoAOD (e.g. FSR radiation)
        ii. Downstream, evaluating DNNs with variated JES is the CPU/GPU bottleneck
    d. Thanks to 100TB of >2GB/s storage, we are CPU-limited :)
4. **Si et al**
    a. Bottleneck is data access for ntuple production, stage-in from various sites to T2, process with grid
    b. Created montly, used daily

Analysis Model Evolution

1. **Lucas et al**
    a. Local production of distributions directly from NanoAOD
    b. CPU-efficiency (RDataFrame) allows even desktop-level processing for iteration
2. **Sergio et al**
    a. Central/grid production of ntuples from NanoAOD, local processing of distributions
3. **Joosep et al**
    a. Local production of distributions directly from NanoAOD
    b. Analysis kernels with CPU/GPU acceleration, local processing
    c. May start to precompute/cache some repeated steps (e.g. prestore JECs like other groups, 10x speed increase)
    d. Foresee with 10x more data: ~10...50TB local storage, 1000 threads / 100 GPUs for a few hours for one full analysis iteration
    e. Processing 1-10% on “laptop/desktop” is feasible
4. **Si et al**
    a. Increased use of low-level inputs in a dynamic fashion

Common

1. Data formats
    a. ~50% use NanoAOD, rest need low-level access
    b. NanoAOD documentation is limited
        i. E.g. does the jet or subjet collection have JECs applied, if so, which ones?
        ii. change logs between versions: private reprocessing due to mistakes or miscommunication
2. Processing
    a. batch queues globally clogged, e.g. 100k jobs for 48h for one analysis
    b. Expert knowledge of grid operations coupled with local support helps to avoid big issues and “ticket hell”
    c. NanoAOD/ntuple production with new branches can be a bottleneck
        i. Remote data reads are a bottleneck before CPUs
        ii. Fast local storage is not available at all sites
        iii. CPU is the limiting factor only when the above are satisfied, but can help a lot
3. Evolution
    a. Local data when possible, CPU-efficient code to make use of it
    b. Temporary staging of data for processing when need to pull from tape or worldwide sites