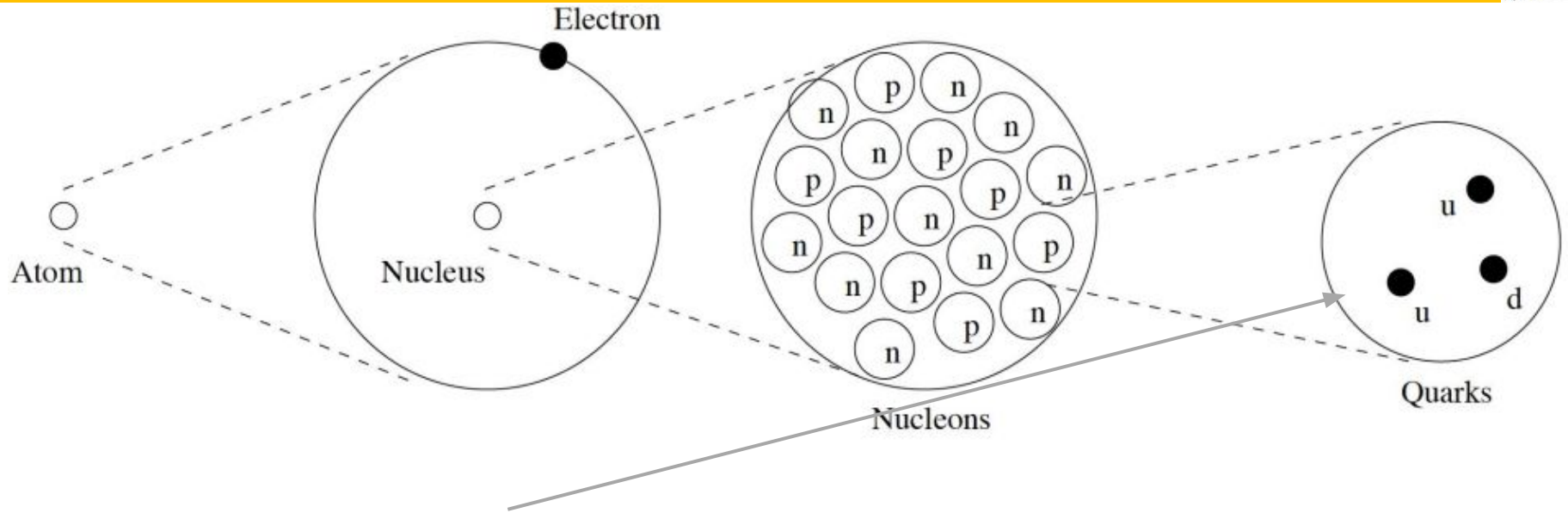
CMS Analyses using subMIT

Guillelmo Gomez-Ceballos (MIT)

Experimental (Collider) Particle Physics



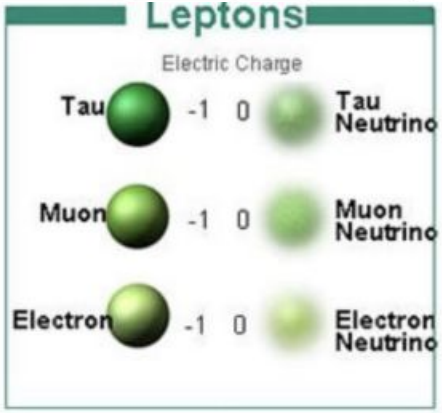
Study of elementary particles and their interactions via collisions of particles

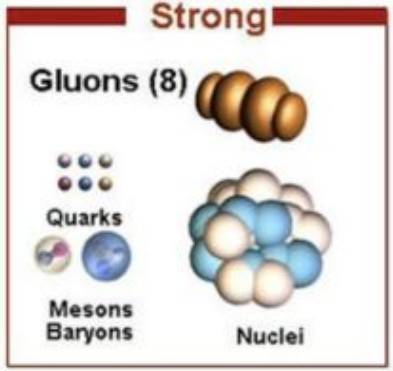
- Large Hadron Collider (LHC): proton-proton collider
- Compact Muon Solenoid (CMS): one of the multi-purpose experiments at LHC

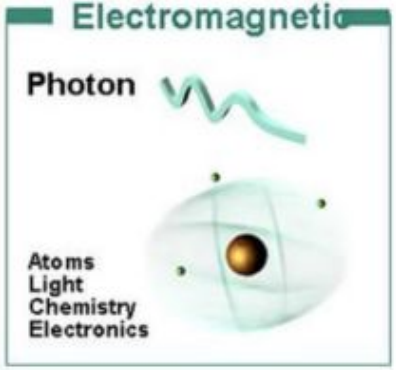
Elementary Particles & Interaction Forces

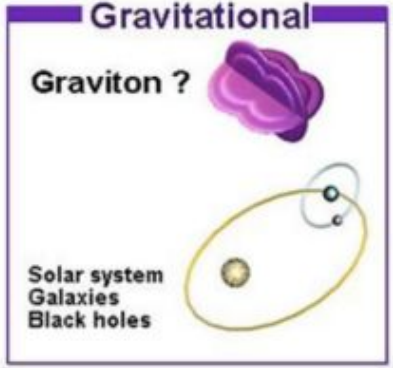
Force	Name	Symbol	Number	EM charge
Strong	Gluons	g	8	0
EM	Photon	γ	1	0
Weak	W and Z	$W^{\pm}$, Z^0	3	± 1, 0

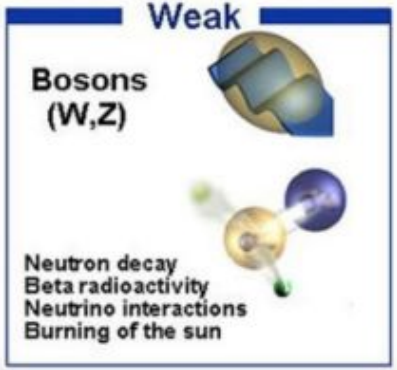
	Generation			Charge	Feels the force of		
	1st	2nd	3rd	Units of e	Strong	EM	Weak
U-Type Quarks (×3 colours)	u	c	t	+2/3	Y	Y	Y
D-Type Quarks (×3 colours)	d	s	b	−1/3	Y	Y	Y
Charged Leptons	e	μ	τ	−1	N	Y	Y
Neutral Leptons (Neutrinos)	ν_e	ν_μ	ν_τ	0	N	N	Y



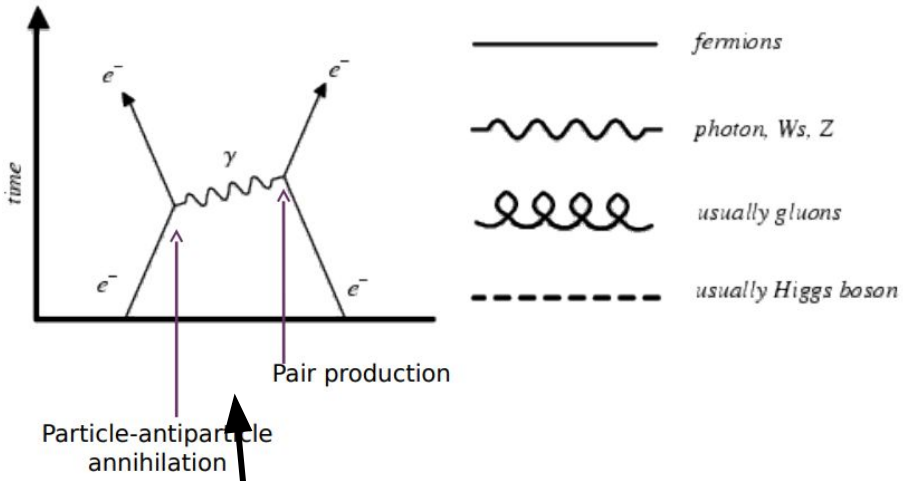


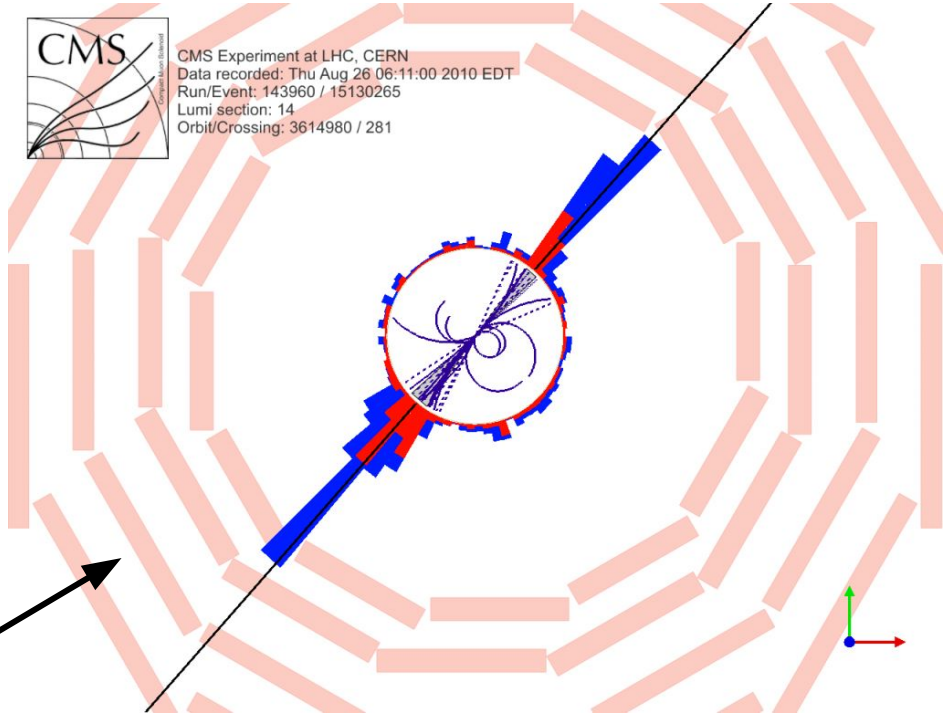






Feynman Diagrams & Particle Collisions



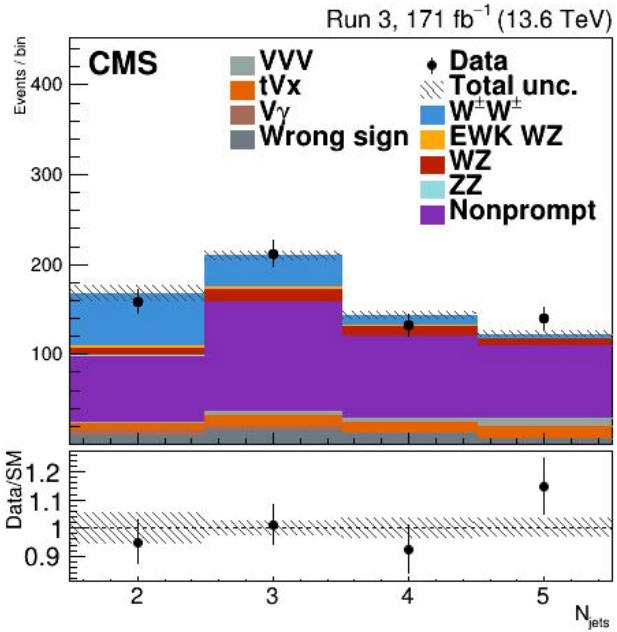


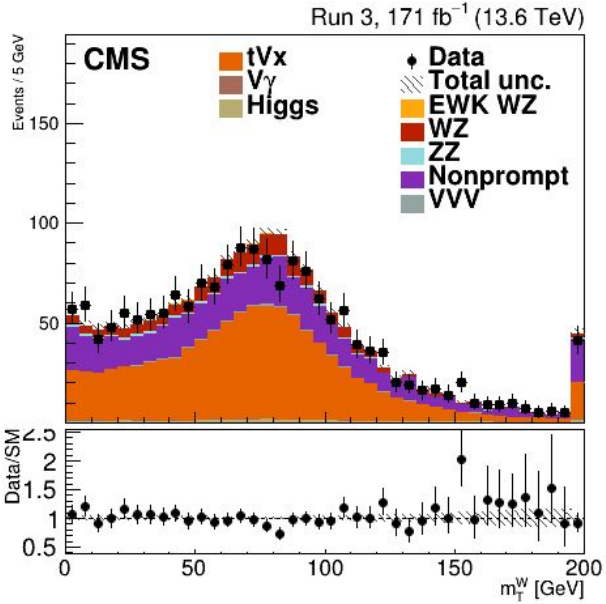
Feynman diagrams vs. real particle collisions

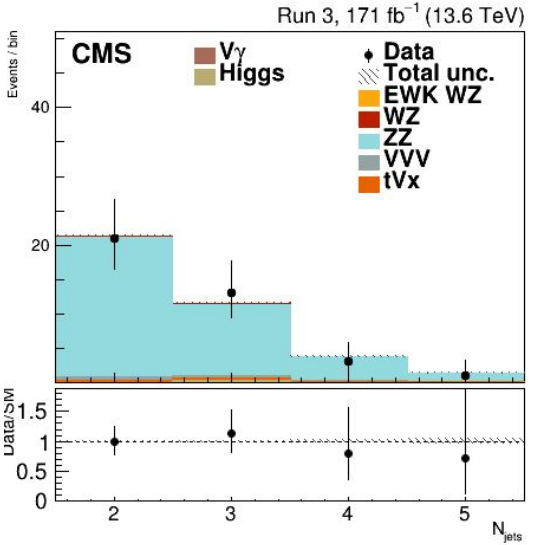
Statistical Analyses

- Need to analyze a lot of (billions) data collisions and compare with the simulated predictions
  - Study the standard model (SM) of particle physics
  - Look for deviations within the SM
  - Directly look for particles or model beyond the SM
- It is not one data and one simulated samples, but many of them
  - Data samples split in time-intervals and years
  - Different simulated samples per physics process
- Several (usual) steps:
  - Online trigger decision: can not take all the data (neither useful nor manageable) → apply similar trigger algorithms to simulation
    - Rejected events gone, critically to make it right
  - Skimming: further reduction the sample size by applying some loose requirements →faster running time
  - Production of output information after a final selection: ``histograms”
  - Final analysis by comparing data and simulation

Some Distributions Using This Framework







Data (dots) compare with predictions (histograms)

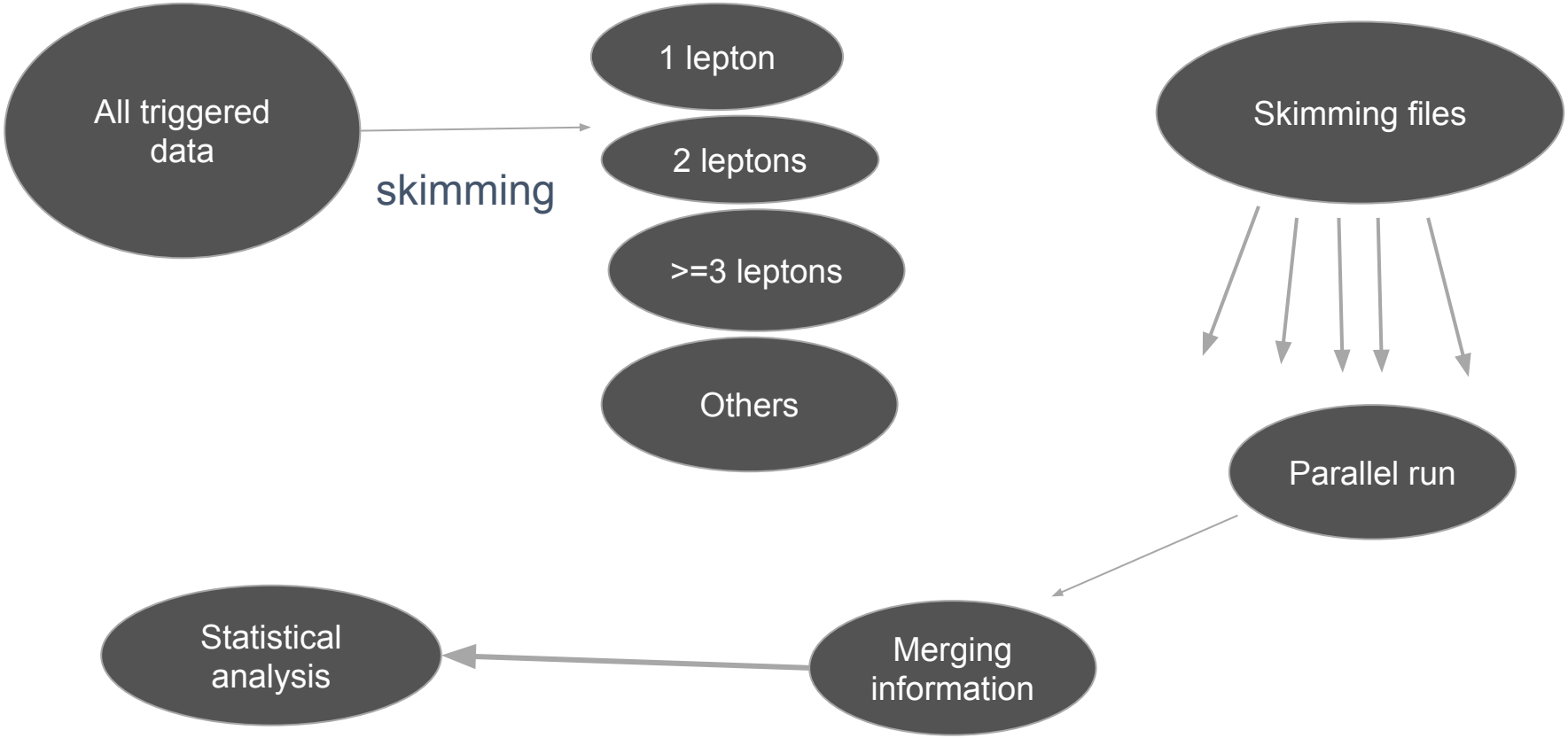
Data Analysis Flow

- CMS analyses targeting multilepton final states
  - Involving a large number of data and MC samples
  - ~60 independent samples X 5 data-taking eras ~ 300 in total
  - Several analyses involved different set of samples
- Purely RootDataFrame (RDF) and NanoAOD based
  - RDF: columnar analysis on ROOT
  - Needs: all within CMSSW (python / ROOT) - CMS Software specific + *libraries*
  - NanoAOD:
    - Relatively small (ROOT-based) samples commonly used in CMS
    - Making use of them out of the box without adding new branches
  - All other needed inputs obtained on-the-fly
    - MC weights, data corrections…
- Possible ``modus operandi”:
  - Run everything at once on a single interactive job using powerful machines (done for W boson mass-related analysis within subMIT)
  - Run interactively with small splitting
  - Parallel running with large splitting

Three Analysis Steps

- Skimming
  - Select events split by 1L, 2L, 3L, MET, γ (one input, five outputs)
  - Jobs submitted to condor on submit
  - Access input samples (worldwide) via xrootd, using global pool
  - Output files on /ceph/submit (potentially on /scratch too)
- Analysis jobs using input skimmed files
  - Common functions for building objects, systematics, weights…
  - RDF jobs running on slurm
  - Split individual samples in N (up to 10) batches
    - 60 X 5 X 4-10 ~ 1200-3000 jobs
    - Most of them run super fast
    - Having a reliable batch system is mandatory
- Studies
  - Merging output histograms/ntuples
  - Set of scripts for measurements, plotting...

Analysis Steps: Graphical Representation



Config Files

- Skimming with condor

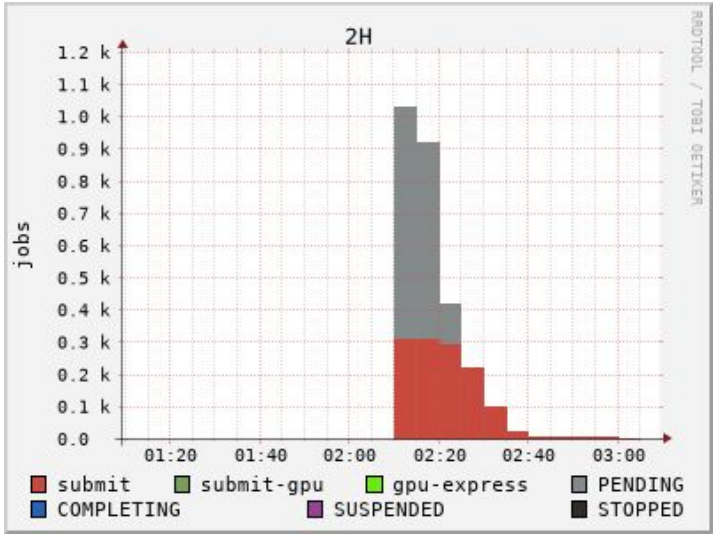
```
cat << EOF > submit
Universe   = vanilla
Executable = skim.sh
Arguments  = ${whichSample} ${whichJob} ${group} skim_input_samples_${YEAR}_fromDAS.cfg skim_input_files_fromDAS.cfg
RequestMemory = 6000
RequestCpus = 1
RequestDisk = DiskUsage
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
transfer_output_files = ""
Log    = logs/simple_skim_${whichSample}_${whichJob}.log
Output = logs/simple_skim_${whichSample}_${whichJob}.out
Error  = logs/simple_skim_${whichSample}_${whichJob}.error
transfer_input_files = skim.tgz, skim_within_singularity.sh
use_x509userproxy = True
x509userproxy = /home/submit/ceballos/x509up_u${USERPROXY}
+AccountingGroup = "analysis.ceballos"
Requirements = ( BOSCOCluster =!= "t3serv008.mit.edu" && BOSCOCluster =!= "ce03.cmsaf.mit.edu" && BOSCOCluster =!= "e
+DESIRED_Sites = "T2_CH_CERN,T2_CH_CERN_AI,T2_CH_CERN_HLT,T2_CH_CERN_Wigner,T2_CH_CSCS,T2_CH_CSCS_HPC,T2_CN_Beijing,T
_IFCA,T2_FI_HIP,T2_FR_CCIN2P3,T2_FR_GRIF_IRFU,T2_FR_GRIF_LLR,T2_FR_IPHC,T2_GR_Ioannina,T2_HU_Budapest,T2_IN_TIFR,T2_I
IR,T2_MY_UPM_BIRUNI,T2_PK_NCP,T2_PL_Swierk,T2_PL_Warsaw,T2_PT_NCG_Lisbon,T2_RU_IHEP,T2_RU_INR,T2_RU_ITEP,T2_RU_JINR,T
CHC,T2_UA_KIPT,T2_UK_London_IC,T2_UK_SGrid_Bristol,T2_UK_SGrid_RALPP,T2_US_Caltech,T2_US_Florida,T2_US_Nebraska,T2_US
CH_CERN_DOMA,T3_CH_CERN_HelixNebula,T3_CH_CERN_HelixNebula_REHA,T3_CH_CMSAtHome,T3_CH_Volunteer,T3_US_HEPCloud,T3_US_
Queue
EOF
```

- Analysis with slurm

```
cat << EOF > submit
#!/bin/bash
#SBATCH --job-name=simple_${whichAna}_${condorJob}_${whichSample}_${whichYear}_${whichJob}
#SBATCH --output=logs/simple_${whichAna}_${condorJob}_${whichSample}_${whichYear}_${whichJob}_%j.out
#SBATCH --error=logs/simple_${whichAna}_${condorJob}_${whichSample}_${whichYear}_${whichJob}_%j.error
#SBATCH --cpus-per-task=4
srun ./analysis_singularity_slurm.sh ${whichSample} ${whichYear} ${whichJob} ${condorJob} ${whichAna}
EOF
```

An Example on Slurm

- Same-sign WW analysis
  - Using 2022-24 samples, 5 data-taking areas
  - Individual skimmed samples small
- 314 individual processes X 4 jobs per sample
  - 1256 jobs in total



- 90% of jobs finished within 20 min
  - ~100 slots available at a given time
- Only 4-6 jobs remained after 30 min

More Optimal?

- In principle, analysis steps could be done faster (?)
  - Given I am working on it on my spare (?) time, and a large number of parallel tasks→not critical
- Accessing files from /scratch faster than /ceph/submit?
- Too many vs. too few split jobs
  - Most samples run very fast →ending time completely dominated by a few jobs
  - Could decide splitting depending on the sample
    - Not an issue as long as enough slots are obtained at a given time
- As a ``back-up" option:
  - Running interactively possible if slurm does not work
    - Slower and unwanted option
    - Sometimes needed when slots are ``blocked" by other users

Summary

- Shown an analysis framework using subMIT
  - By no means this is maybe the most optimal approach, but it works
- A reliable computing system absolutely critical
  - Note there are several independent analyses which need to be run in parallel
  - One iteration for all analyses may involved ~6k slurm jobs
- Good performance overall speaking
  - Feedback and help from support have been great so far!