Using Big Data Technologies for HEP Analysis

The CMS Big Data Project:

M. Cremonesi, O. Gutsche, B. Jayatilaka, J. Kowalkowski, S. Sehrish **[FNAL]**

L. Canali, V. Dimakopoulos, M. Girone, V. Khristenko, E. Motesnitsalis **[CERN-IT]**

S.-Y. Hoh, J. Pazzini, M. Zanetti **[Padova]**

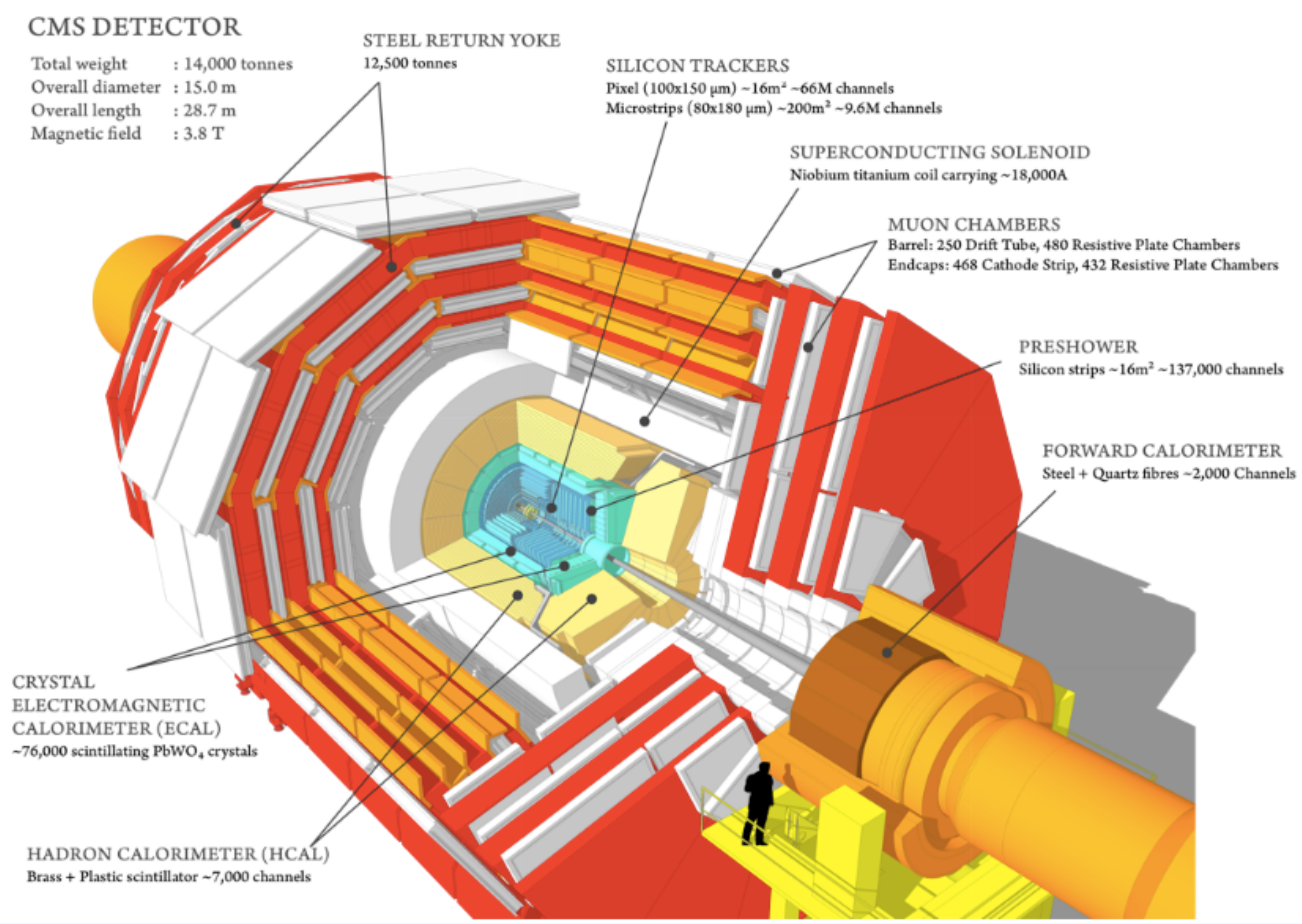
P. Elmer, J. Pivarski, A. Svyatkovskiy **[Princeton]**

I. Fisk **[Simons Foundation]**

A. Melo **[Vanderbilt]**

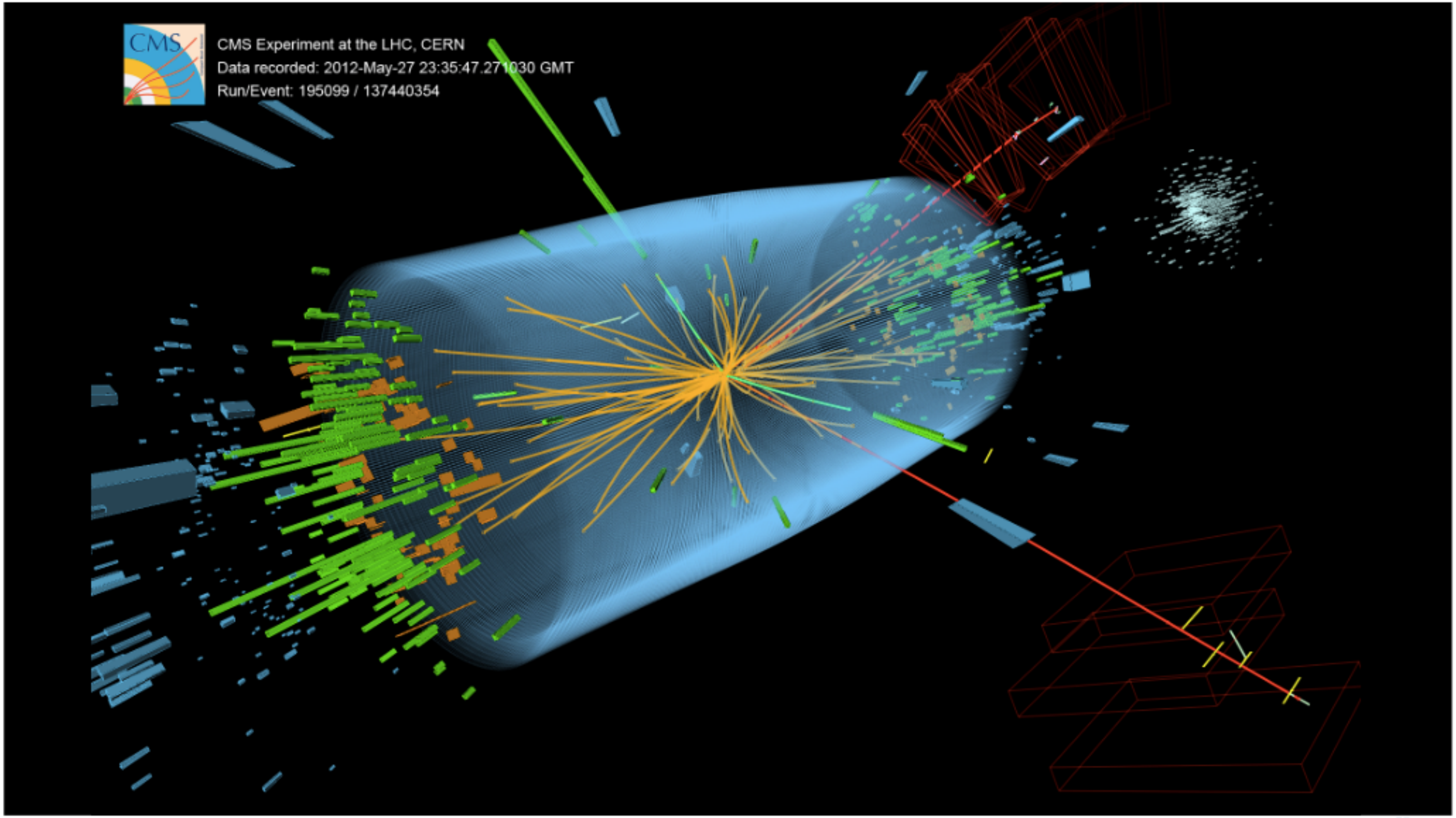
CHEP, 10 July 2018

Data Events

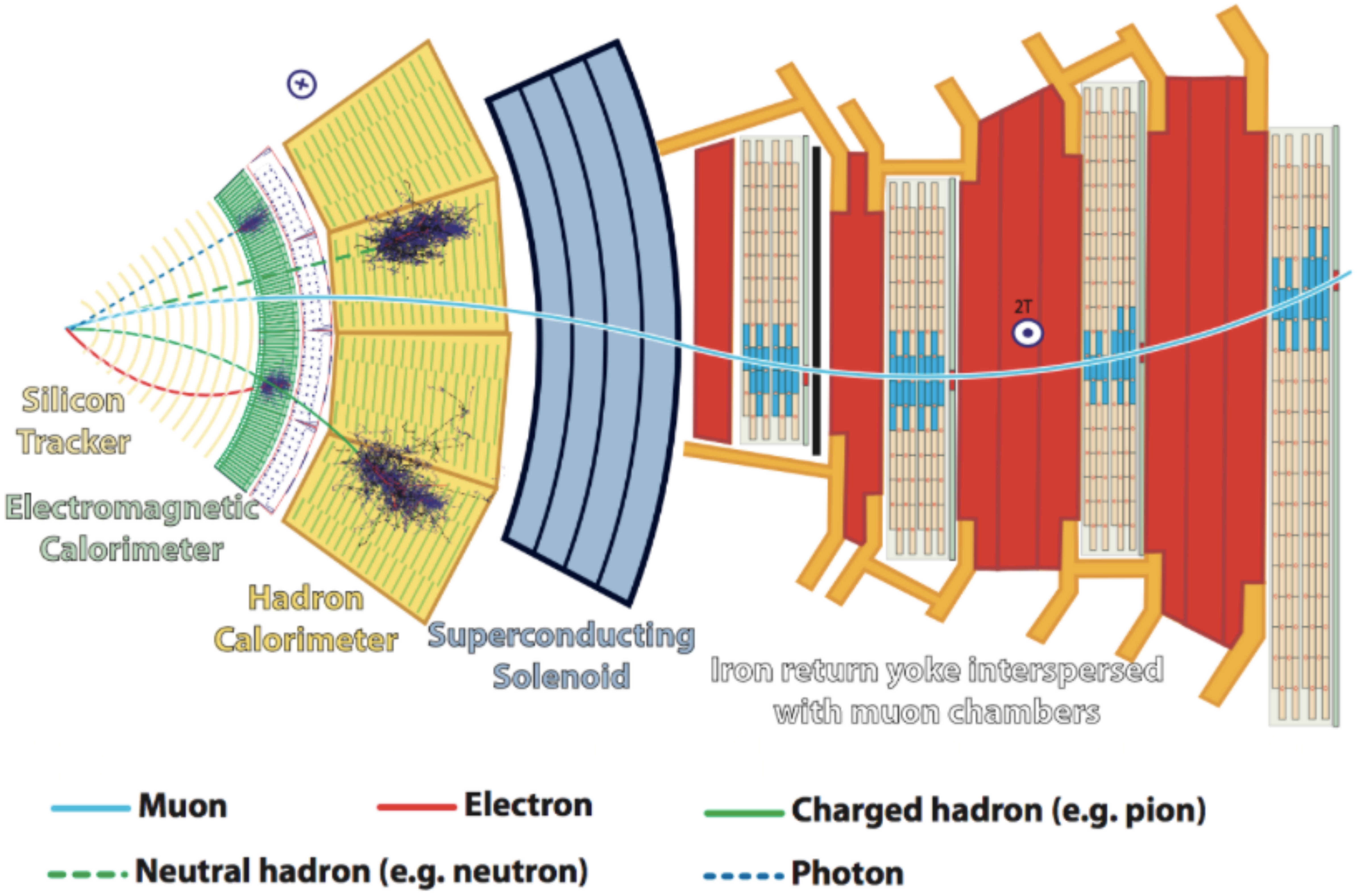


- **Particle detection** = record physics quantities (energy, flight path) of particles produced in a collision
  - Quantities **measured** from the interaction of particles and the different detector components
    - 100 Million individual measurements
  - All measurements of a collision together are called **event**

Fermilab


CMS Experiment at the LHC, CERN
Data recorded: 2012-May-27 23:35:47.271030 GMT
Run/Event: 195099 / 137440354

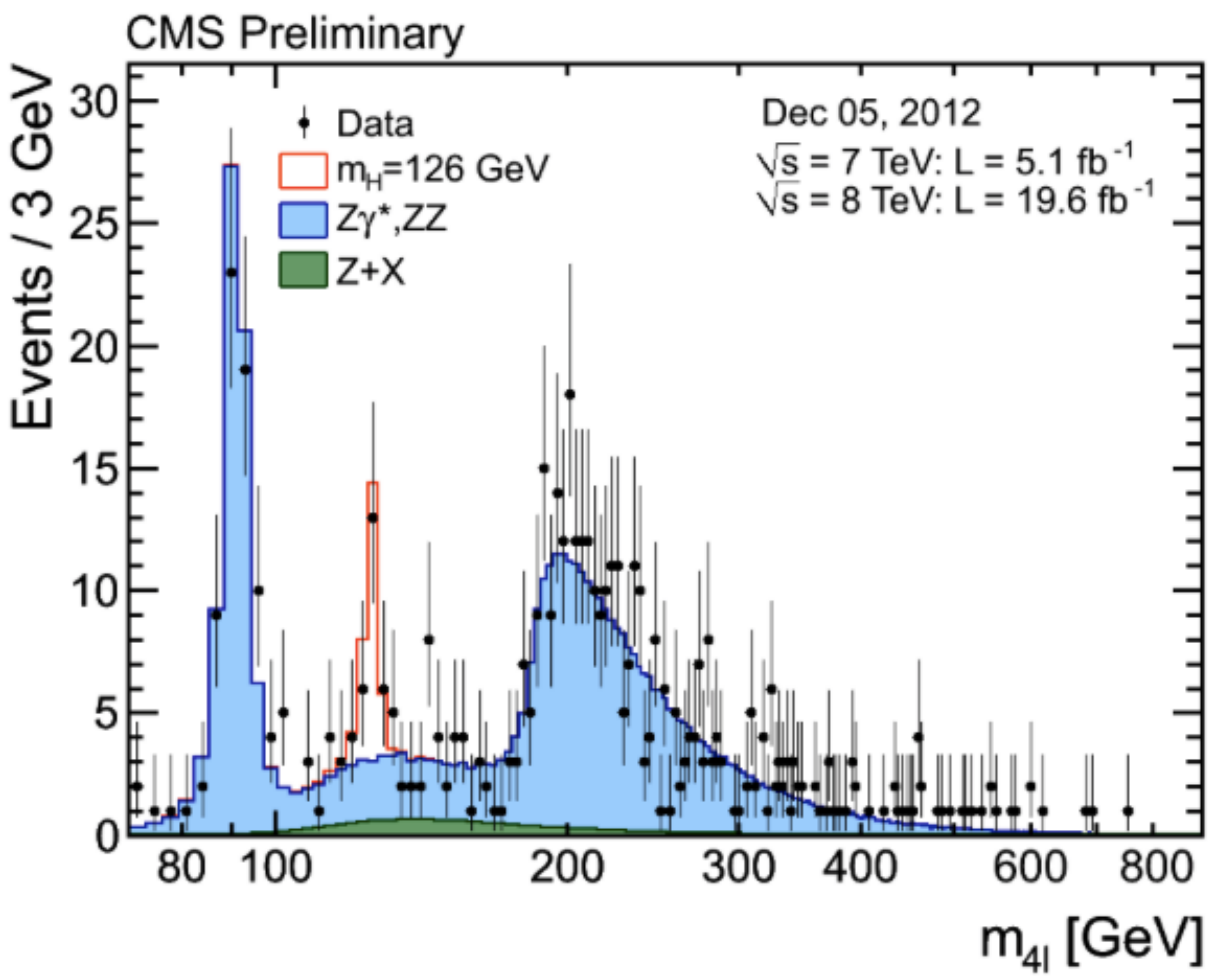
Event Reconstruction



- Detector signals (and equivalent simulated signals) need to be reconstructed to learn about the particles that produced them
- The reconstructed events are then used for analysis

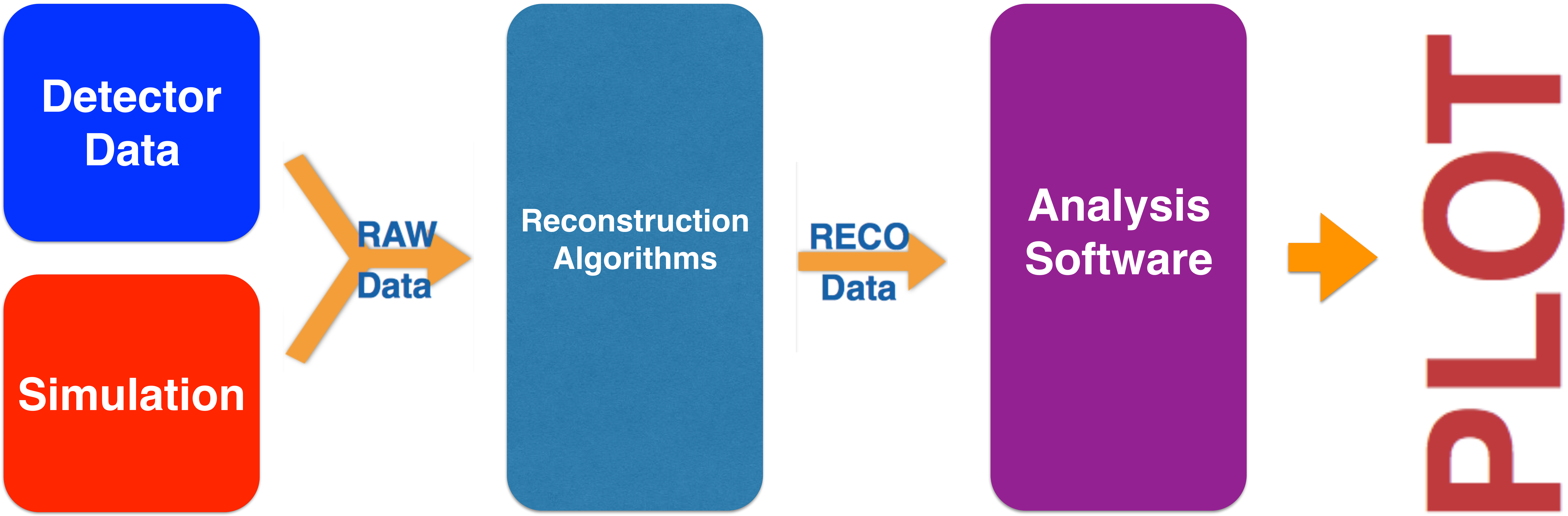
Detector → DAQ & Trigger → Software & Computing → SCIENCE

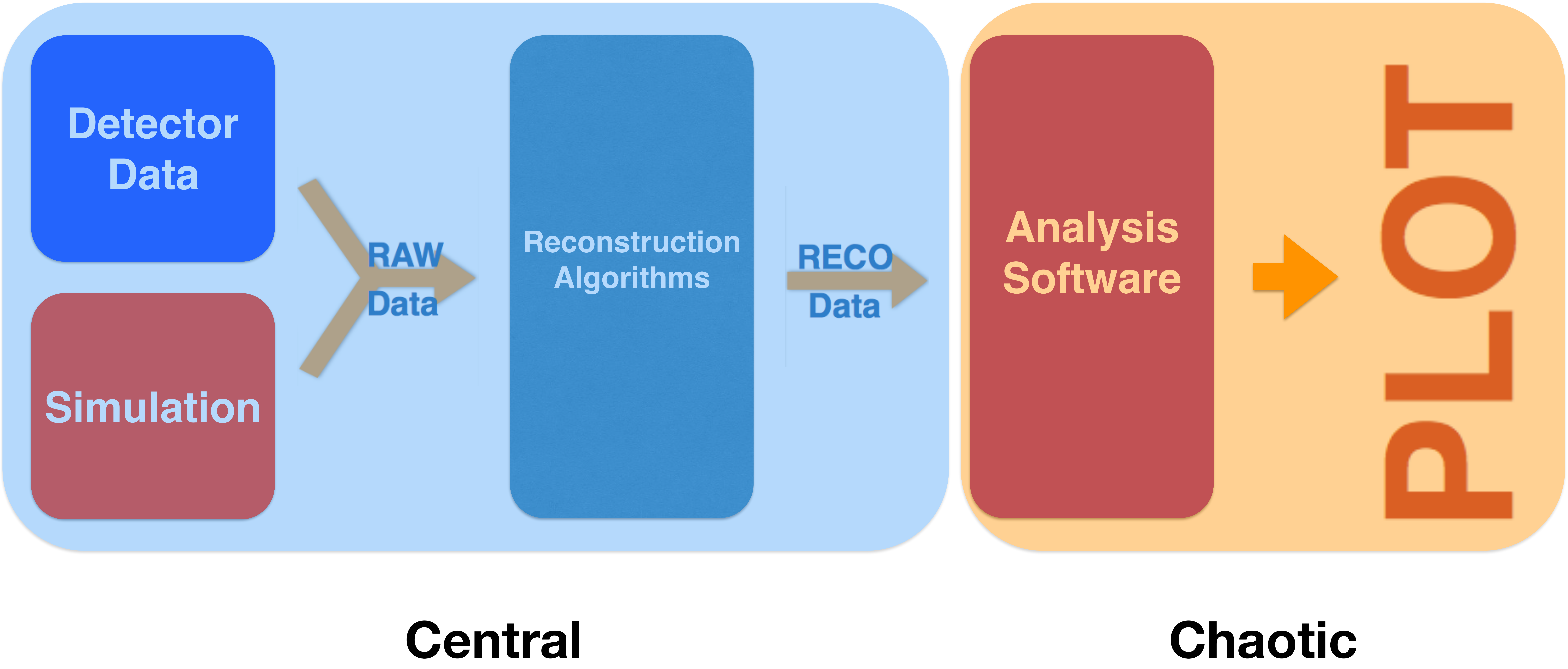
Experimental Particle Physics from Computing Perspective



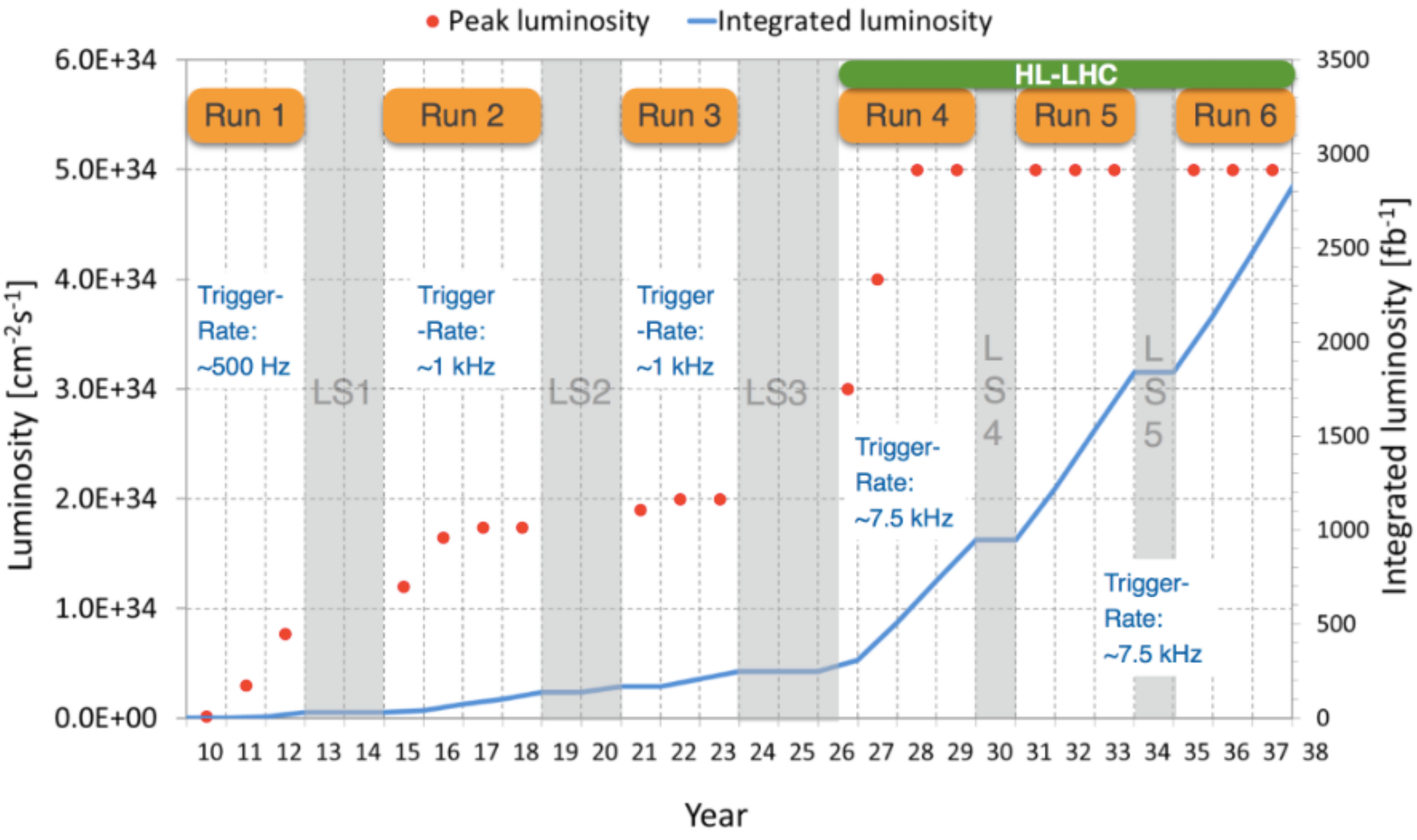
- Detect particle interactions (data), compare with theory predictions (simulation)
  - **Black dots: recorded data**
  - **Blue shape: simulation**
  - **Red shape: simulation of new theory (in this case the Higgs)**

Fermilab


Detector Data
Simulation
RAW Data
Reconstruction Algorithms
RECO Data
Analysis Software
PLOT



Detector Data
Simulation
RAW Data
Reconstruction Algorithms
RECO Data
Analysis Software
PLOT
Central
Chaotic

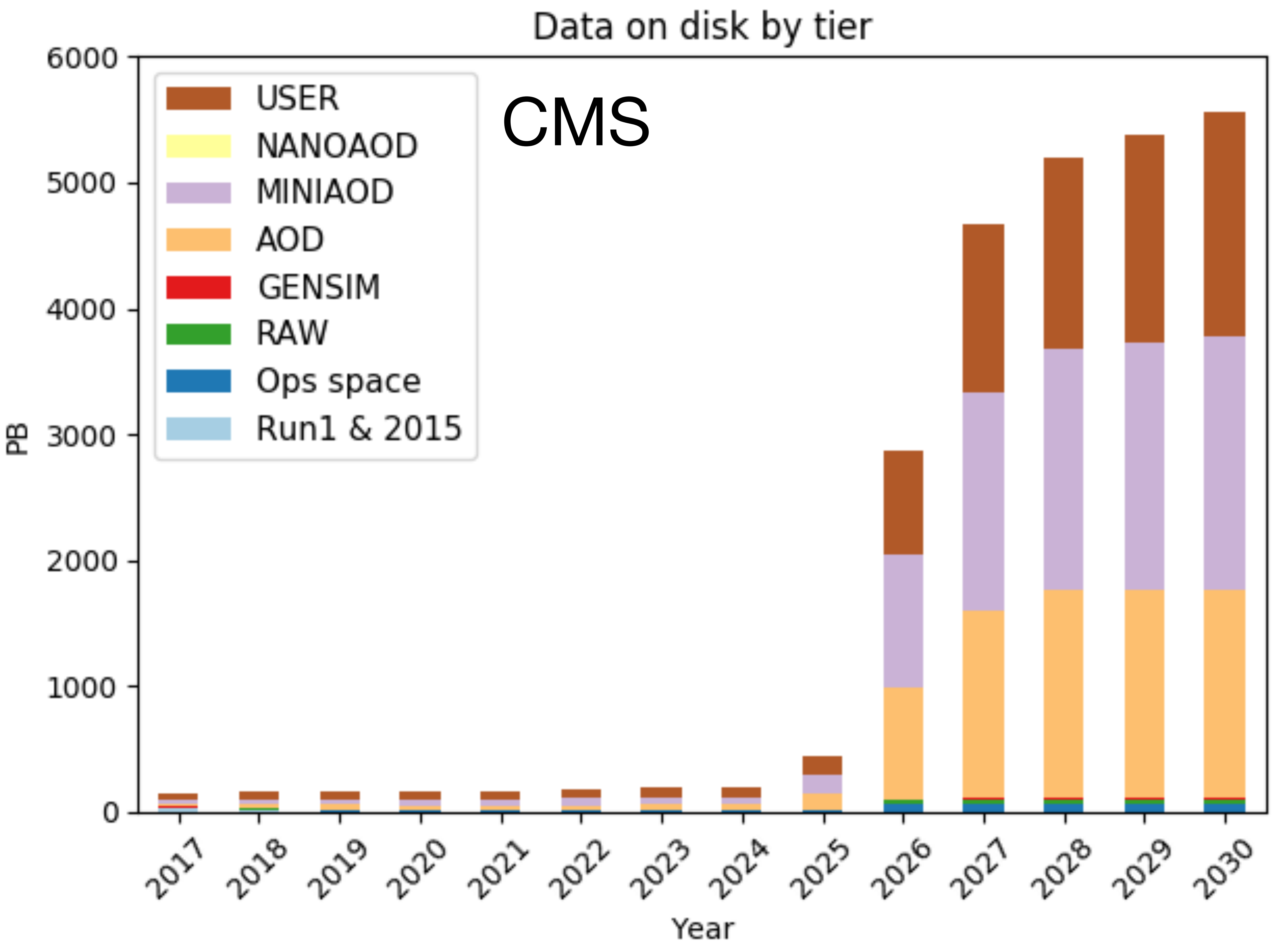


Peak luminosity
Integrated luminosity
HL-LHC
Run 1
Run 2
Run 3
Run 4
Run 5
Run 6
LS1
LS2
LS3
LS4
LS5
Trigger-Rate: ~500 Hz
Trigger-Rate: ~1 kHz
Trigger-Rate: ~1 kHz
Trigger-Rate: ~7.5 kHz
Trigger-Rate: ~7.5 kHz
Luminosity [cm-2s-1]
Integrated luminosity [fb-1]
Year
6.0E+34
5.0E+34
4.0E+34
3.0E+34
2.0E+34
1.0E+34
0.0E+00
3500
3000
2500
2000
1500
1000
500
0
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38

Fermilab

CMS Data Volume @ HL-LHC

- Extract physics results will require to handle/analyze a lot more data
  - must trim inefficiencies
- Explore industry technologies as suitable candidates for user analysis



CMS Big Data Project

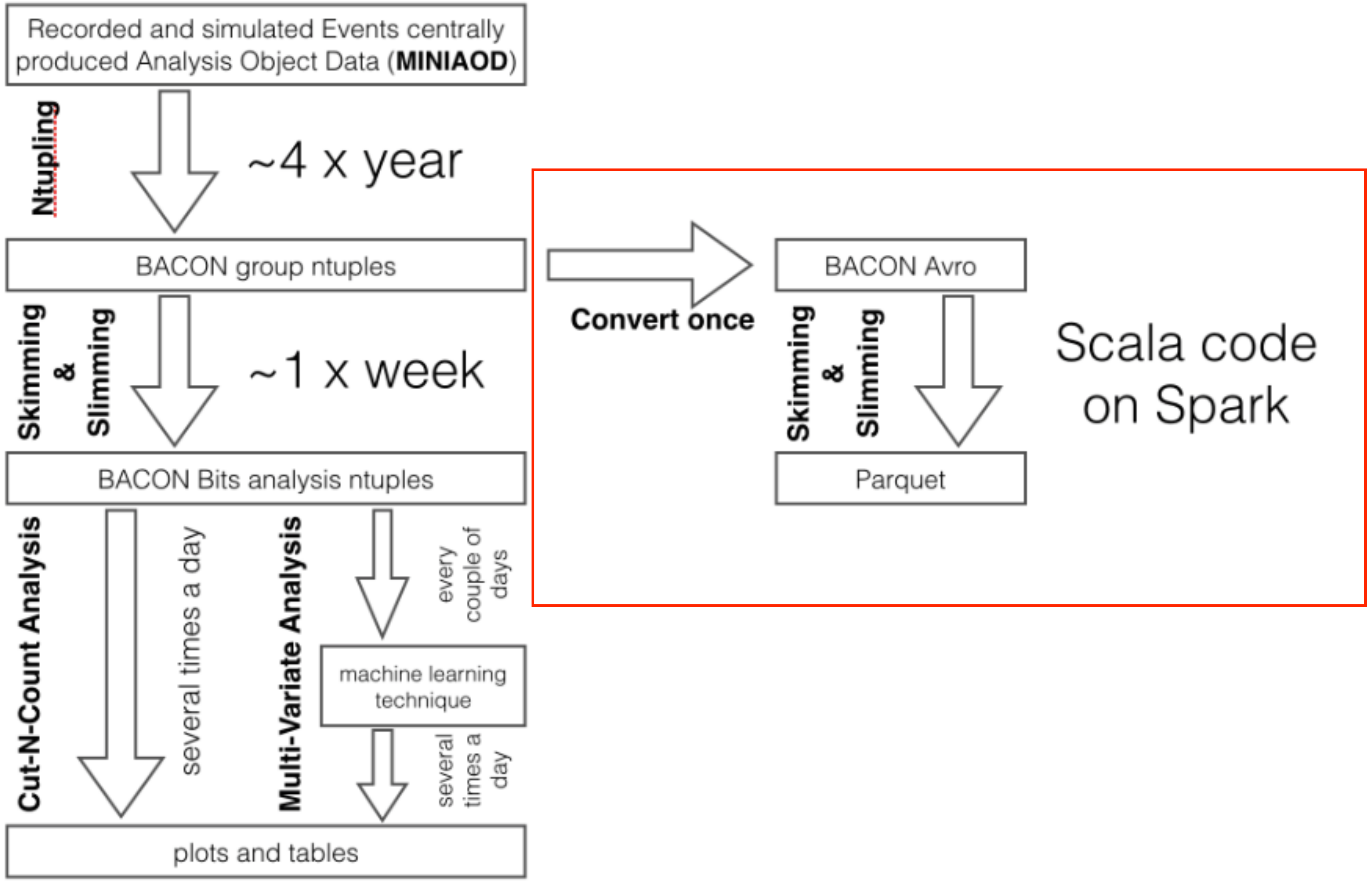
- Group created end of 2015
  - collaboration between FNAL, Diana-HEP, and CERN-IT
  - website: https://cms-big-data.github.io
- Rapidly expanding:
  - Vanderbilt and Padova joined last year
- CERN Openlab enables partnership with industry:
  - CERN Openlab/Intel project called "CMS Data Reduction Facility"
  - Project includes CERN fellow supporting the development and testing of the reduction facility
  - Intel actively taking part in project
  - Sponsoring of CERN fellow included in the project

Thanks to Intel and Cofluent for the support over these years

CHEP 2016: Proof of Principle

arXiv:1711.00375

- Usability Study using Apache Spark:
  - Analyzer code in Scala
  - Input converted in Avro: https://github.com/diana-hep/rootconverter
- Improved user experience with optimized bookkeeping



Fermilab

ACAT 2017: Steps Forward

arXiv:1703.04171

Several technical advancements:

- **stability to read root files in Spark**: https://github.com/diana-hep/spark-root, eliminating the need to convert in a more suitable format
- **Capability to read input files remotely using XRootD** (e.g. from EOS at CERN): https://github.com/cerndb/hadoop-xrootd , eliminating the need to store files on HDFS

Fermilab

Outline

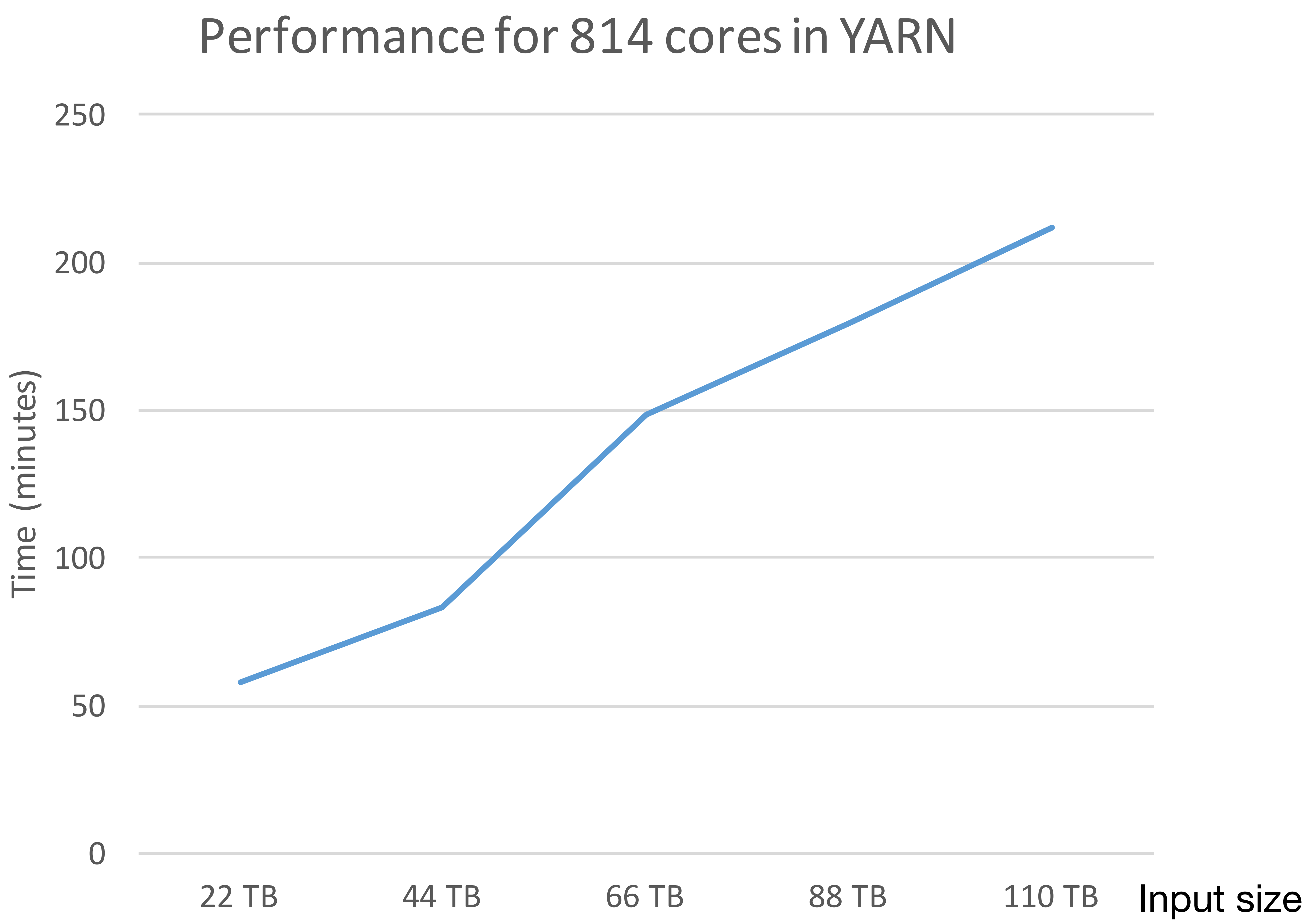
- Scalability tests and first performance measurements
  - Test the capability to reduce 1 PB of data to 1 TB in less than five hours with the new tools developed by CERN-IT
- Review of real analysis use cases in Apache Spark
  - Tools developed by CERN IT applied at Padova and Vanderbilt to real physics analysis
  - Usability test and current limitation
- What’s next
  - Goal for the next year

Scalability Tests, Infrastructure and Workload

- Spark cluster:
  - **analytix @ CERN**: shared infrastructure with **~1300 cores, 7 TB RAM**
- Storage:
  - HDFS and Remote EOS Public/UAT
- Simple physics analysis use case is applied to select events and reduce the datasets

Scalability Test/1

Increasing the input size while maintaining the same amount of resources

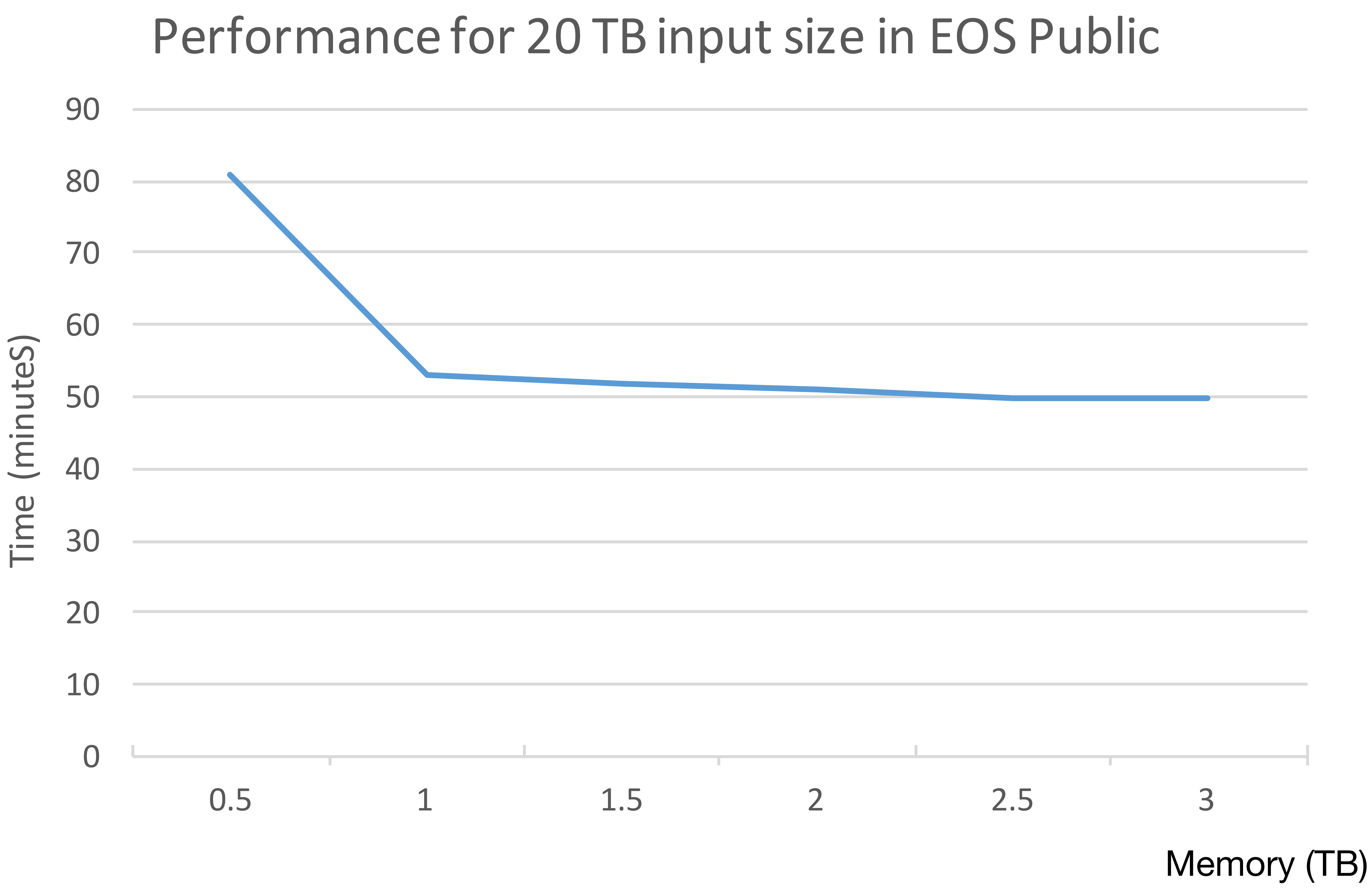


Input Data	Time for EOS Public
22 TB	58m
44 TB	83m
66 TB	149m
88 TB	180m
110 TB	212m

Initial configuration: 407 executors, 2 vcores per executor, 7 GB per executor

Scalability Test/2

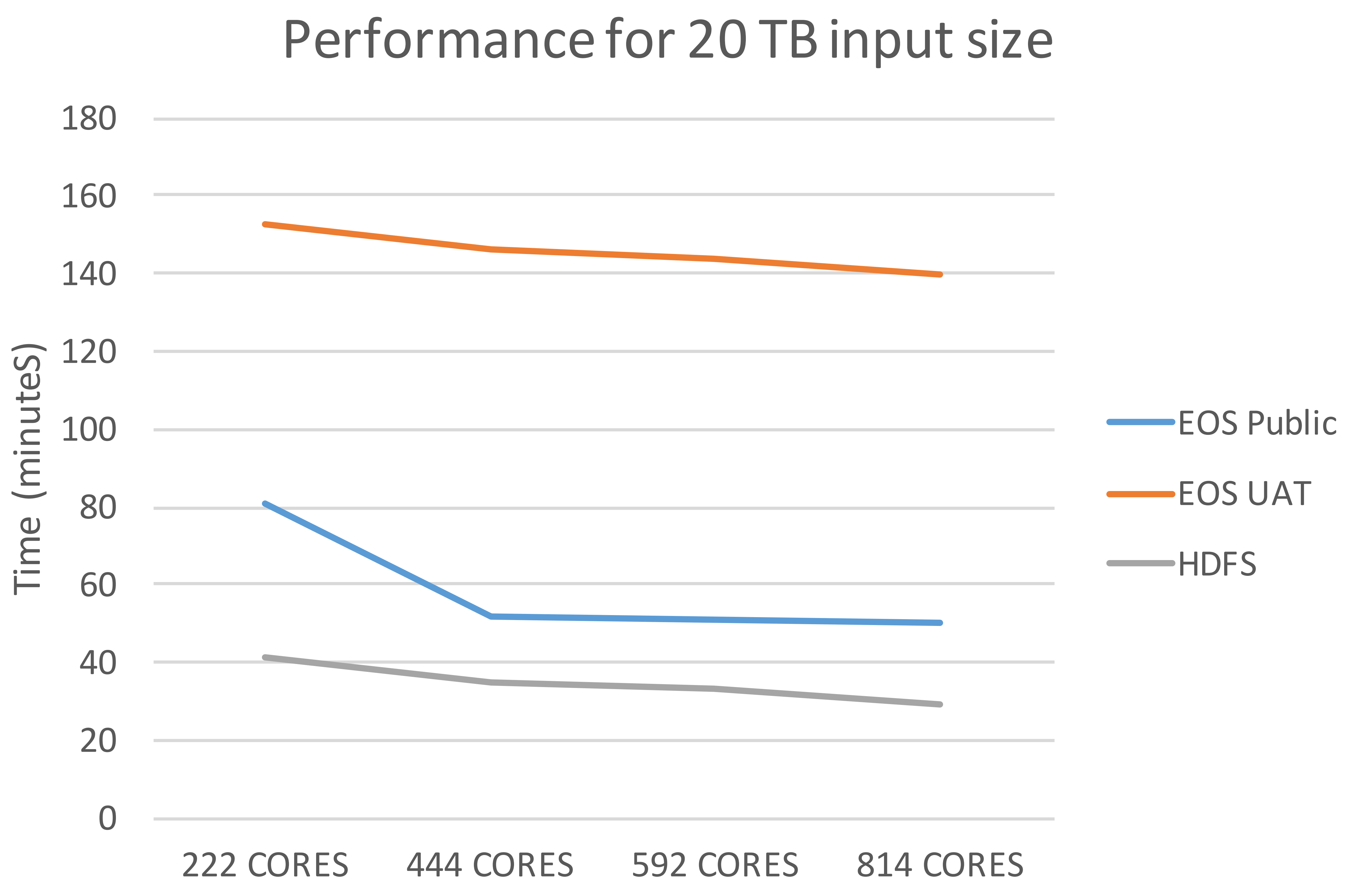
Increasing the resources while maintaining the same input size (for 2:1 vcore-executor ratio)



Number of Executors/Cores	Total Memory:	Runtime:
74/148	0.5 TB	81m
148/296	1 TB	53m
222/444	1.5 TB	52m
296/592	2 TB	51m
370/740	2.5 TB	50m
444/888	3 TB	50m

Scalability Test/3

Comparing EOS Public, EOS UAT, and HDFS (191, 6, and 38 nodes respectively)

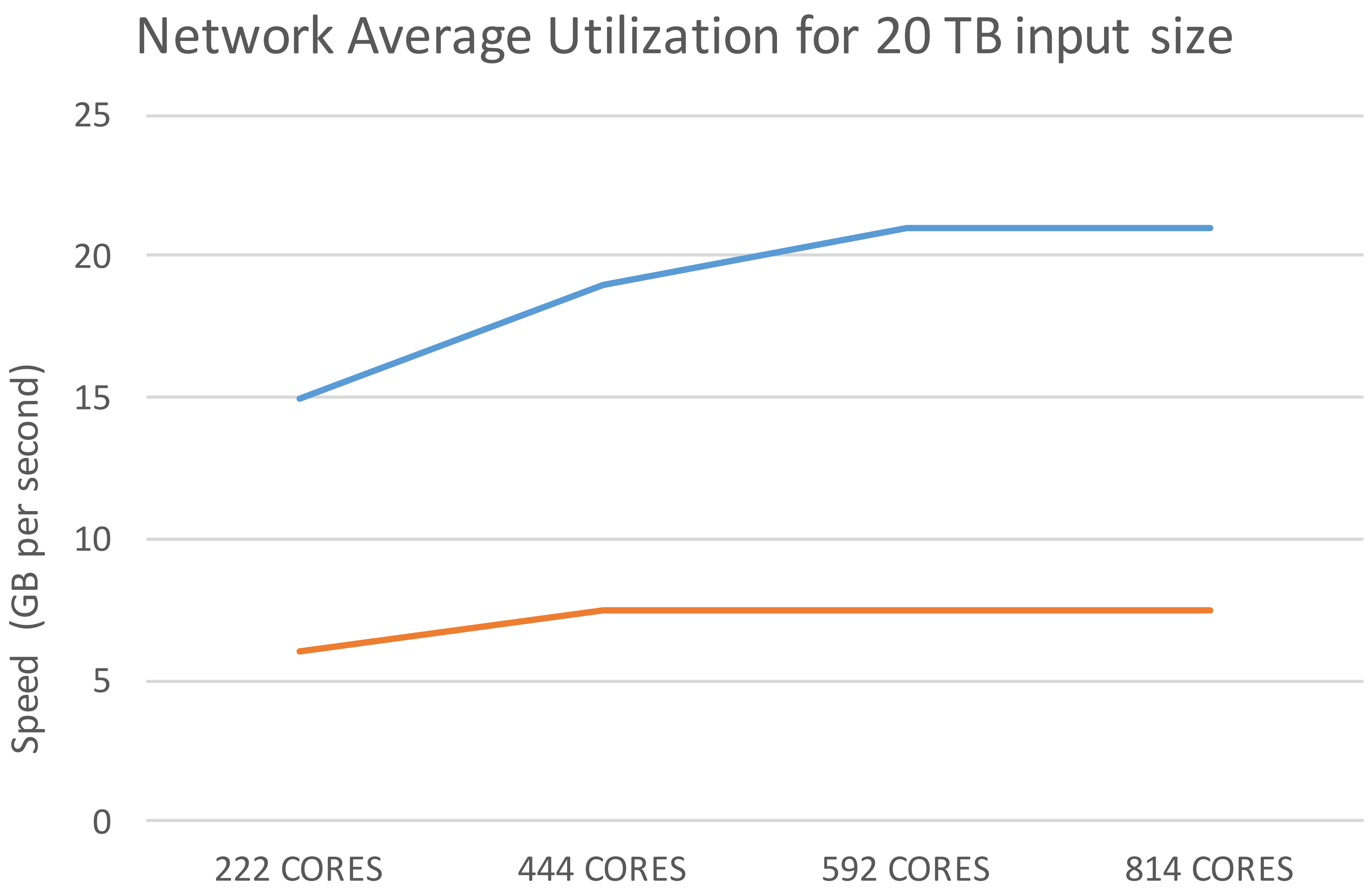


Number of Executors/ VCores:	Runtime for EOS Public:	Runtime for EOS UAT:	Runtime for HDFS:
111/222	81m	153m	41m
222/444	52m	146m	35m
296/592	51m	144m	33m
407/814	50m	140m	29m

Fermilab

Scalability Test/4

Understanding the bottlenecks - high network load



Cores:	EOS Public	EOS UAT
222 vcores	15 Gbytes/s	6 Gbytes/s
444 vcores	19 Gbytes/s	7.5 Gbytes/s
592 vcores	21 Gbytes/s	7.5 Gbytes/s
814 vcores	21 Gbytes/s	7.5 Gbytes/s

Fermilab

Observations

- We can reduce Data with 72 TB/h
  - Current bottleneck: network throughput from remote storage to Spark cluster
  - We measured up to 20 Gigabytes/s throughput to EOS Public
- These results are about a factor 3 from our original goal of reducing 1 PB to 5 hours
  - Reasonably done with more hardware or software optimizations (Work In Progress)
- Workload optimization profited from cooperation with Intel with Intel CoFluent Technology
  - For this particular job, optimal results were obtained at the 2:1 vcore-executor ratio

Analysis Use-case @ Vanderbilt/Padova

Analysis workflow:

- Load standard ROOT files as DataFrames (DFs)
- Open files over XRootD
- Use Spark to transform DFs
- Aggregate DFs into histograms
- Produce plots, tables, etc.. from histograms

Tools used:

- Spark-ROOT - ROOT in Spark
- Hadoop-XRootD - XRootD FS support for Hadoop
- Histogrammar - Data aggregation
- Matplotlib - Python-based plotting

Identical physics use cases, using similar strategy, same tools, but different infrastructure

Usability Test

- Make a first-year CS undergraduate student run the workflow
  - No knowledge of physics whatsoever, limited computing knowledge
  - Able to make the Vanderbilt workflow run in one day
- **Portability**
  - Run the Padova code at Vanderbilt
  - Major showstopper: environment setup

=> need to write sort of a shared library with site configuration towards **full generalization**

What's Next: Coffea Development

- Generalized version of the code used so far by Padova/Vanderbilt
  - Fully portable (no configuration issues)
  - Use-case independent (in principle it already is)
- **CO**mpact **F**ramework **F**or **E**laborate **A**lgorithms
- Consist in:
  - List of centrally-produced dataset in experiment-specific format needed for analysis => **coffeabeans**
  - Custom-made version of the experiment software to produce privately datasets in the experiment-specific format => **CoffeaGrinder** (this step may be needed to add information)
  - List of privately/centrally produced dataset in experiment-specific format => **coffeapowder**
  - Apache Spark analysis code => **CoffeaMaker**
  - Reduced datasets/analysis plots => **coffeacups**
  - Interface with the experiment statistical packages => **CoffeaDrinker**

Conclusion

- 2016: proof of principle of the usage of big data technologies in HEP analysis
  - Limitation have been identified to set the focus for 2017
- 2017: development of new tools
  - Spark-root, Hadoop-XRootD connector

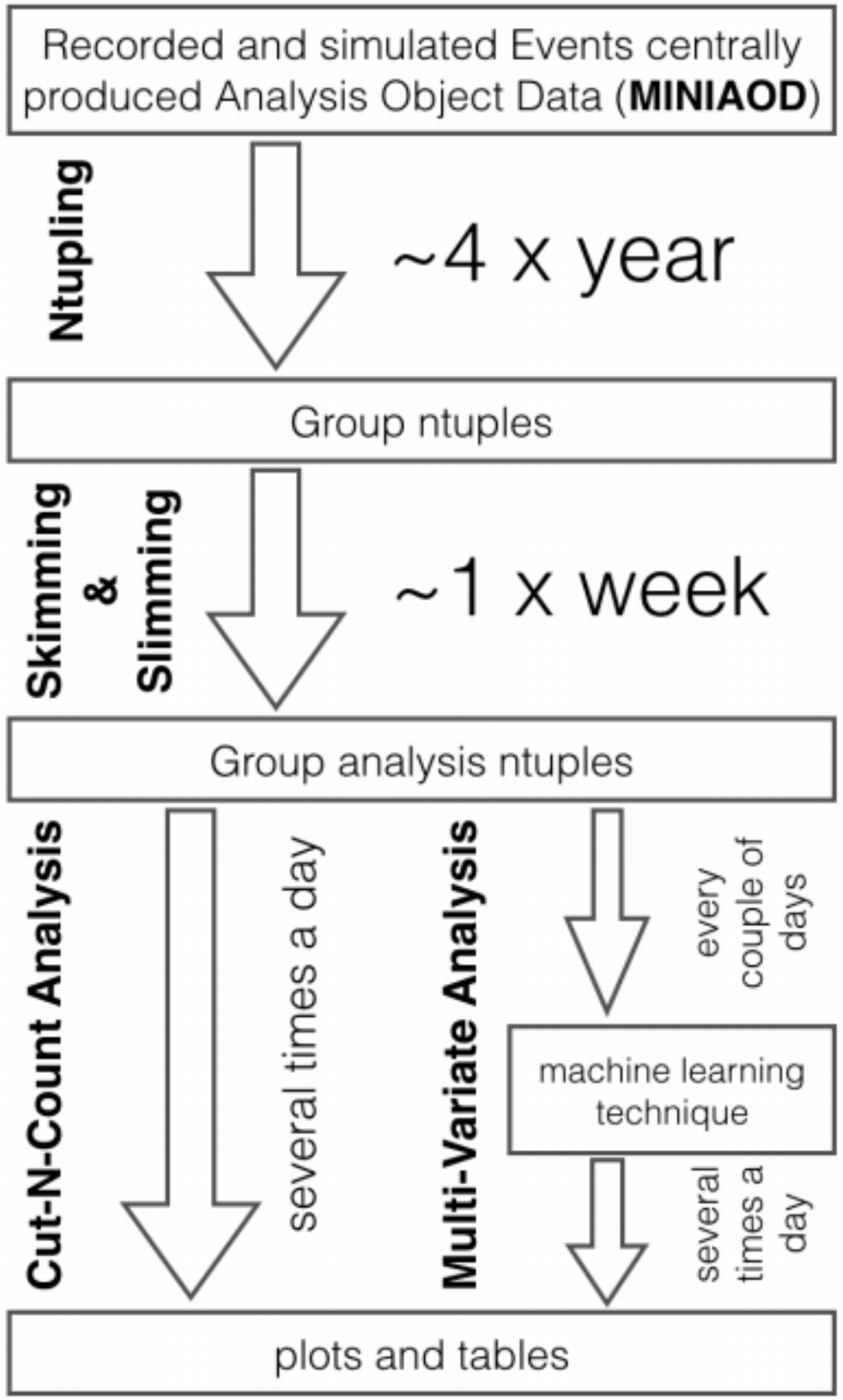
You are here

- **2018: scalability and usability tests, performance measurements**
  - Some bottlenecks have been identified
    - Scalability test: need to scale up the Spark infrastructure and possibly the network
    - Usability test: need to generalize the site configuration
- 2019: Coffea development

Backup

Current Analysis Workflow

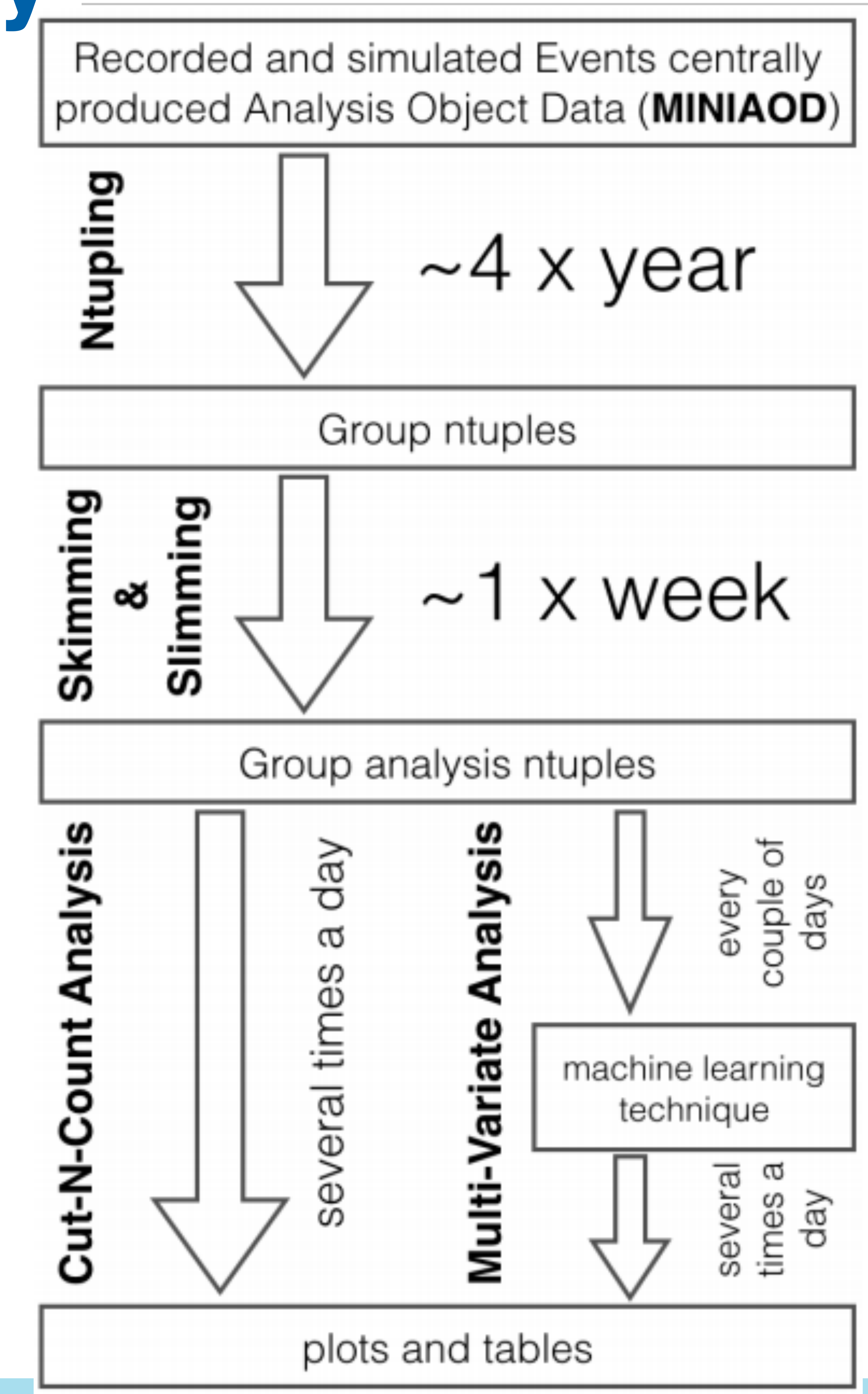
- **Input**:
  - Centrally produced output of reconstruction software, reduced content optimized for analysis
  - Apply updated CMS reconstruction recipes
  - Too big for interactive analysis
- **Ntupling**:
  - Convert into format suited for interactive analysis
  - Still too big for interactive analysis
- **Skimming & Slimming**:
  - dropping events/branches in a disk-to-disk copy
- **Filtering & Pruning**:
  - selectively reading events/branches into memory



Fermilab

CMS Data Reduction Facility

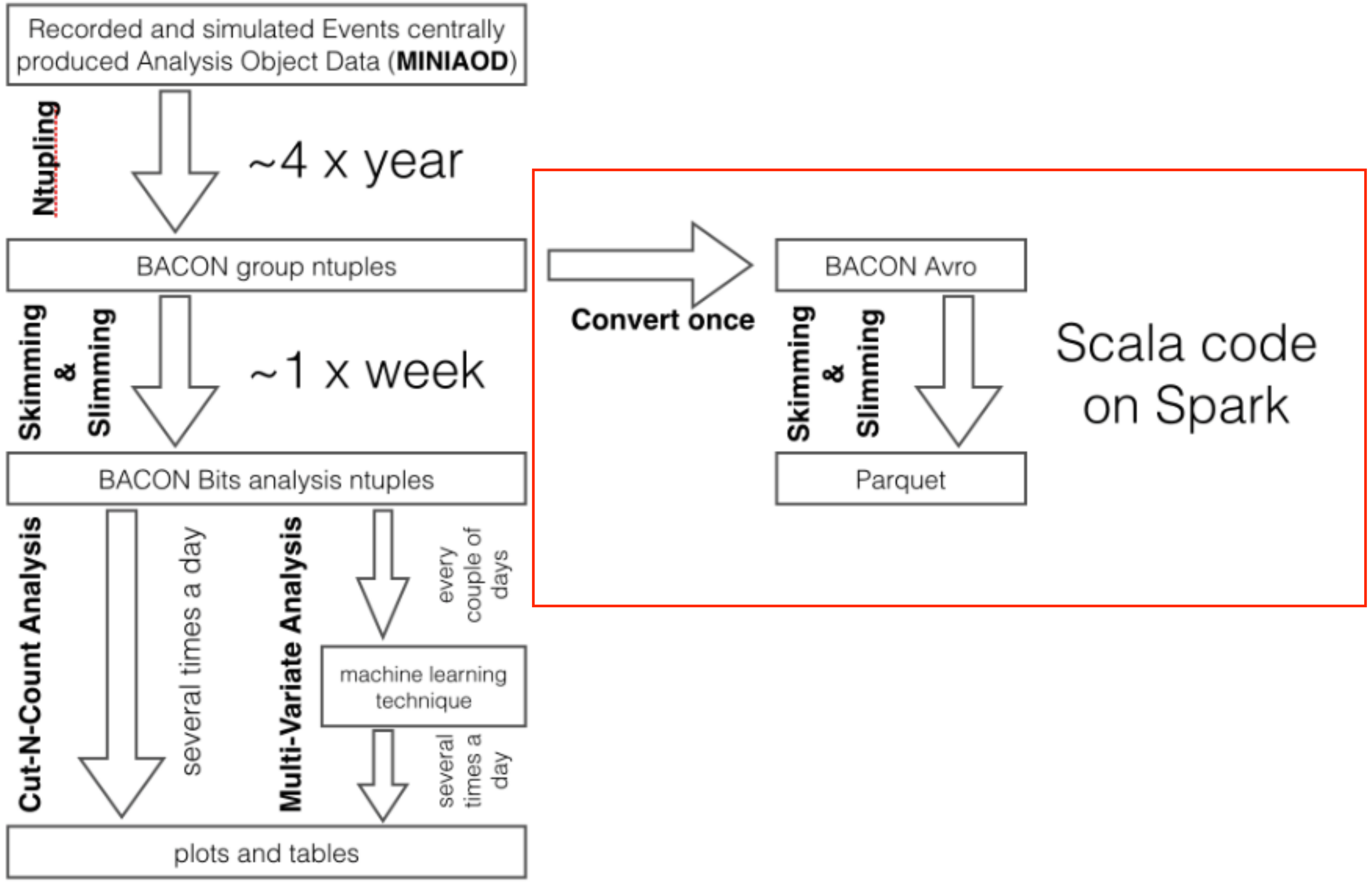
- CERN openlab / Intel project
- Demonstrate reduction capabilities producing analysis ntuples using Apache Spark
- Goal: reduce 1 PB input to 1 TB output in 5 hours (CERN Openlab/ Intel project)



Fermilab

CHEP 2016: Proof of Principle

- Not changing the analysis workflow, optimizing the **bookkeeping**
  - Apache spark
  - Analyzer code in Scala
  - Input converted in Avro: https://github.com/diana-hep/rootconverter, stored on the HDFS



Fermilab

Two loops over file entries, parallel jobs in Spark across cluster

```
// Reference the whole dataset (not individual files)
val mcsample = avrordd("hdfs://path/to/mcsample/*.avro")

// First pass (and cache for later)
mcsample.persist()
val mc_sumOfWeights = mcsample.map(_.GenInfo.weight).sum

// Second pass on data in cluster's memory
val result = mcsample.filter(cuts).map(toNtuple(_, mc_sumOfWeights, mc_xsec))

// Save as ntuple
result.toDF().write.parquet("hdfs://path/to/mcsample_ntuple")
```

Input

Sum of Weights for Simulation

Main Event Selection

Output

Output ntuple is used for analysis e.g: plots, fits, tables

```
# Bring the ntuple in as a DataFrame
ntuple = spark.read.parquet("hdfs://path/to/mcsample_ntuple")

ntuple.select("mass").show()
...
```

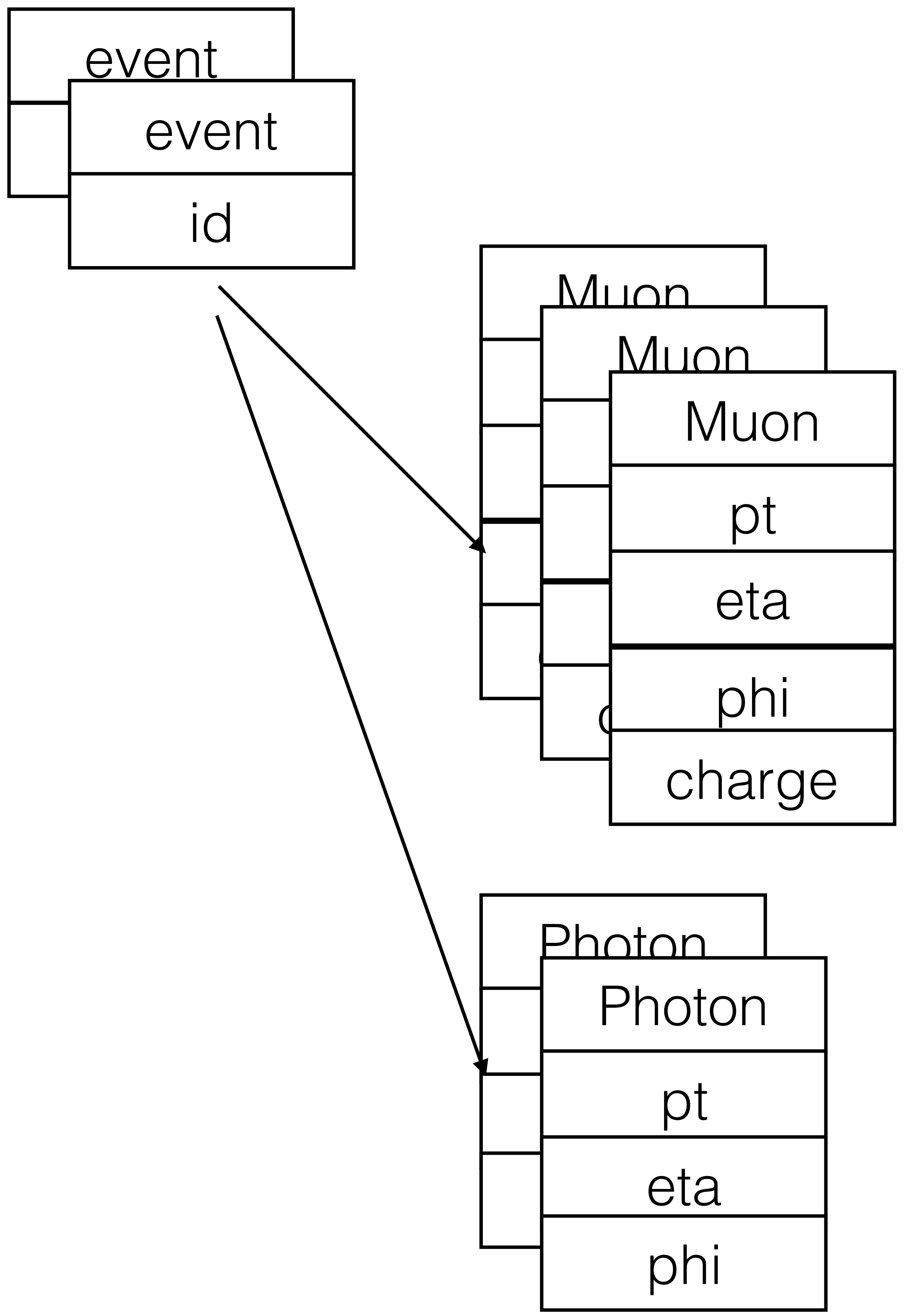
Physics plots!

Output contains information of:

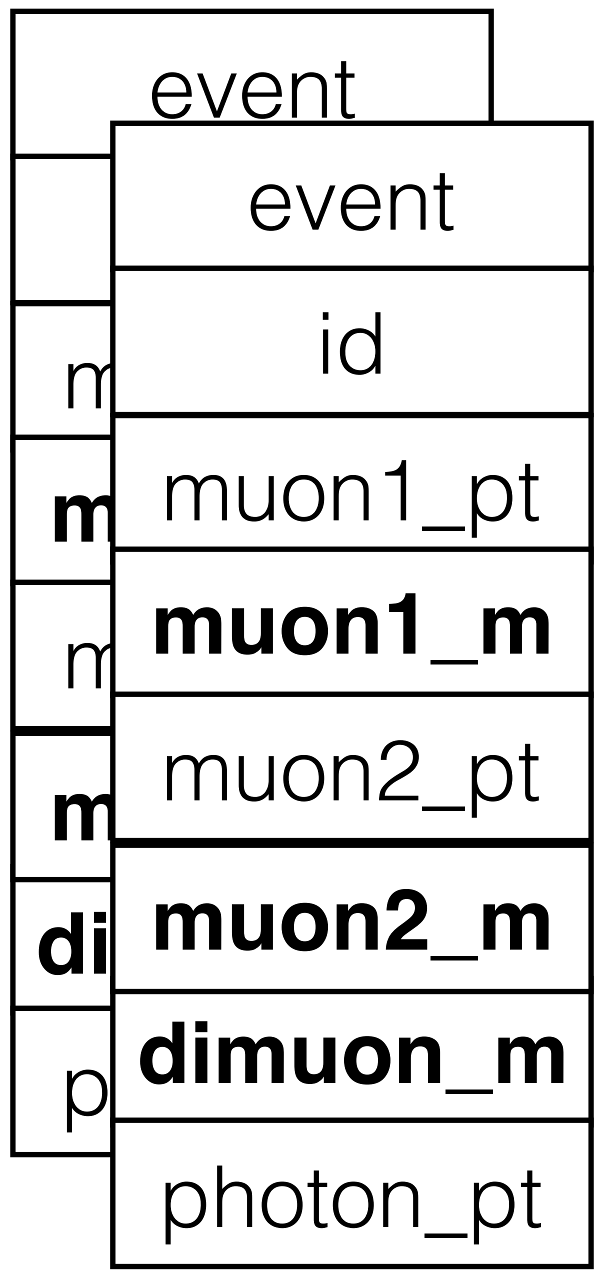
- Object (e.g. Muon/Jet)
- Event (e.g. Luminosity) information

Fermilab

MiniAOD



Analysis Ntuple



```
from PandaCore.Tools.Misc import *
from re import sub

metTrigger='(trigger&1)!=0'
eleTrigger='(trigger&2)!=0'
phoTrigger='(trigger&4)!=0'

metFilter='metFilter==1 && egmFilter==1'
presel = 'nFatjet==1 && fj1Pt>200 && nTau==0 && Sum$(jetPt>30 && jetIso)<2'

cuts = {
 'signal'  : tAND(metFilter,tAND(presel,'nLooseLep==0 && nLooseElectron==0 && nLoosePhoton==0 && pfmet>200 && dphipfmet>0.4')),
 'mn'      : tAND(metFilter,tAND(presel,'nLoosePhoton==0 && nTau==0 && nLooseLep==1 && looseLep1IsTight==1 && abs(looseLep1PdgId)==13 && pfUWmag>200 && dphipfUW>0.4 && mT<160')),
 'en'      : tAND(metFilter,tAND(presel,'nLoosePhoton==0 && nTau==0 && nLooseLep==1 && looseLep1IsTight==1 && looseLep1IsHLTSafe==1 && abs(looseLep1PdgId)==11 && pfmet>50 &&
pfUWmag>200 && dphipfUW>0.4 && mT<160')),
 'zmm'     : tAND(metFilter,tAND(presel,'pfUZmag>200 && dphipfUZ>0.4 && nLooseElectron==0 && nLoosePhoton==0 && nTau==0 && nLooseMuon==2 && nTightLep>0 && 60<diLepMass &&
diLepMass<120')),
 'zee'     : tAND(metFilter,tAND(presel,'pfUZmag>200 && dphipfUZ>0.4 && nLooseMuon==0 && nLoosePhoton==0 && nTau==0 && nLooseElectron==2 && nTightLep>0 && 60<diLepMass &&
diLepMass<120')),
}

for r in ['mn','en']:
        cuts['w'+r] = tAND(cuts[r],'isojetNBtags==0')
        cuts['t'+r]     = tAND(cuts[r],'isojetNBtags==1')

for r in ['signal','zmm','zee']:
        cuts[r] = tAND(cuts[r],'isojetNBtags==0')

for r in ['signal','wmn','tmn','wen','ten','zmm','zee']:
        cuts[r] = tAND(cuts[r],'fj1DoubleCSV>0.75')
        cuts[r+'_fail'] = tAND(cuts[r],'fj1DoubleCSV<=0.75')

weights = {
  'signal'         : '%f*sf_pu*sf_tt*normalizedWeight*sf_lepID*sf_lepIso*sf_lepTrack*sf_ewkV*sf_qcdV*sf_metTrig*sf_btag0',
  'top'            : '%f*sf_pu*sf_tt*normalizedWeight*sf_lepID*sf_lepIso*sf_lepTrack*sf_ewkV*sf_qcdV*sf_btag1',
  'w'              : '%f*sf_pu*sf_tt*normalizedWeight*sf_lepID*sf_lepIso*sf_lepTrack*sf_ewkV*sf_qcdV*sf_btag0',
  'z'              : '%f*sf_pu*sf_tt*normalizedWeight*sf_lepID*sf_lepIso*sf_lepTrack*sf_ewkV*sf_qcdV*sf_btag0',
#  'photon'         : '%f*sf_pu*normalizedWeight*sf_ewkV*sf_qcdV*sf_pho*sf_phoTrig *sf_qcdV2j*sf_btag0', # add the additional 2-jet kfactor
}
weights['qcd'] = weights['signal']
weights['signal_fail'] = weights['signal']
```

Fermilab