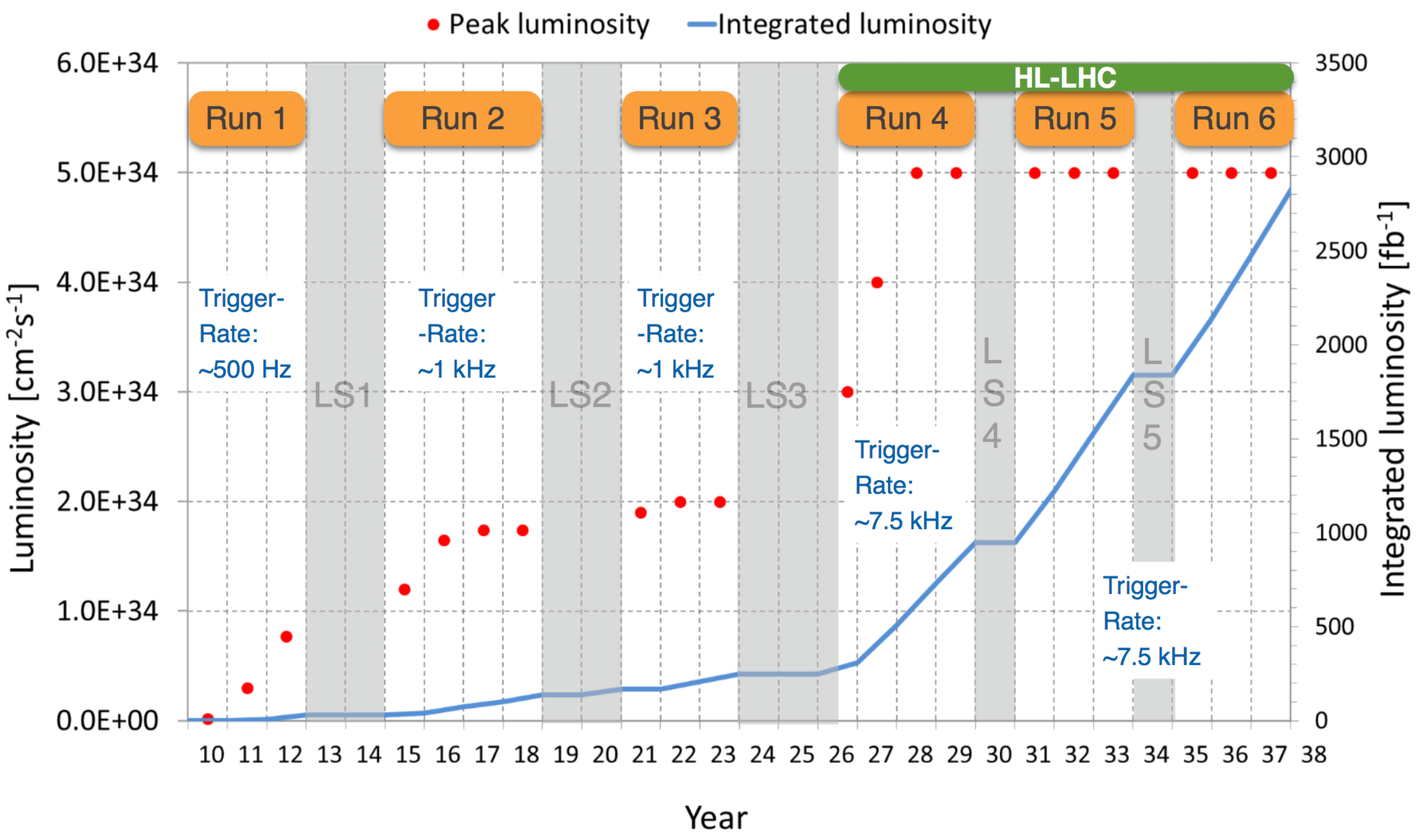
“Big Data” in HEP: A comprehensive use case study

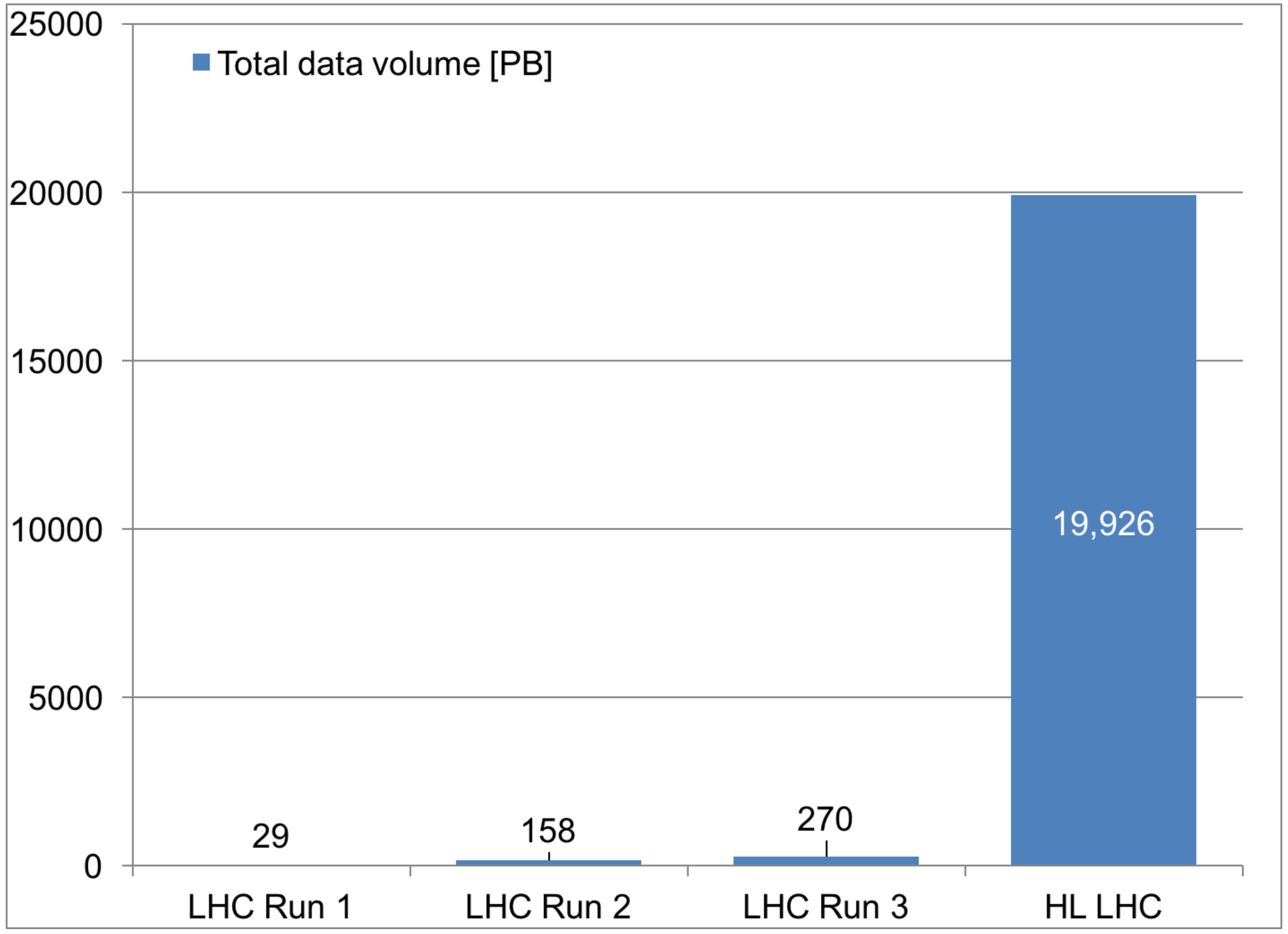
Oliver Gutsche, Matteo Cremonesi,, Bo Jayatilaka, Jim Kowalkowski, Saba Sehrish, Cristina Mantilla Suárez, Nhan Tran - Fermi National Accelerator Laboratory

Peter Elmer, Jim Pivarski, Alexey Svyatkovskiy - Princeton University

22nd International Conference on Computing in High energy and Nuclear Physics, 10.-14. October 2016

The HL-LHC challenge



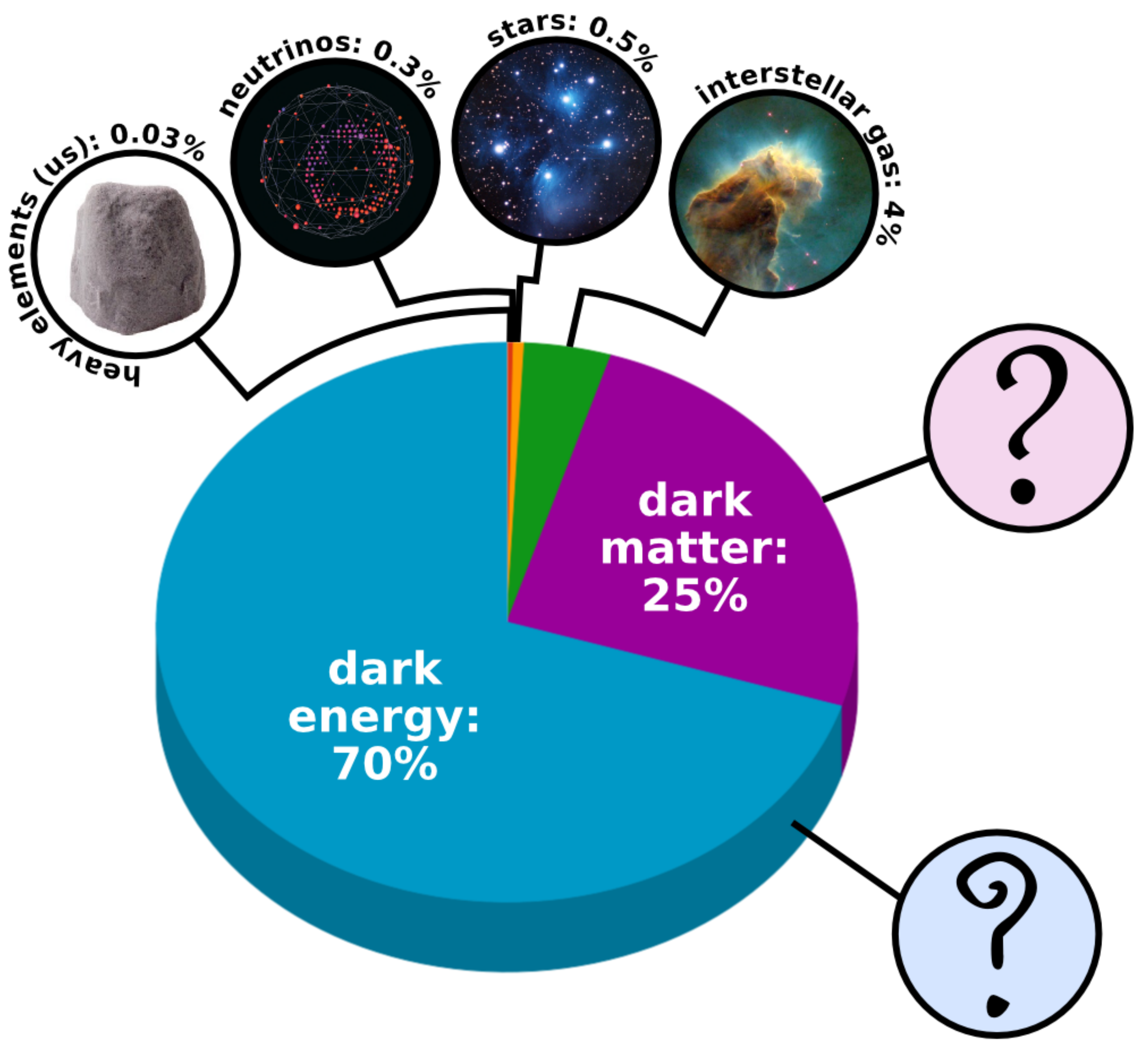


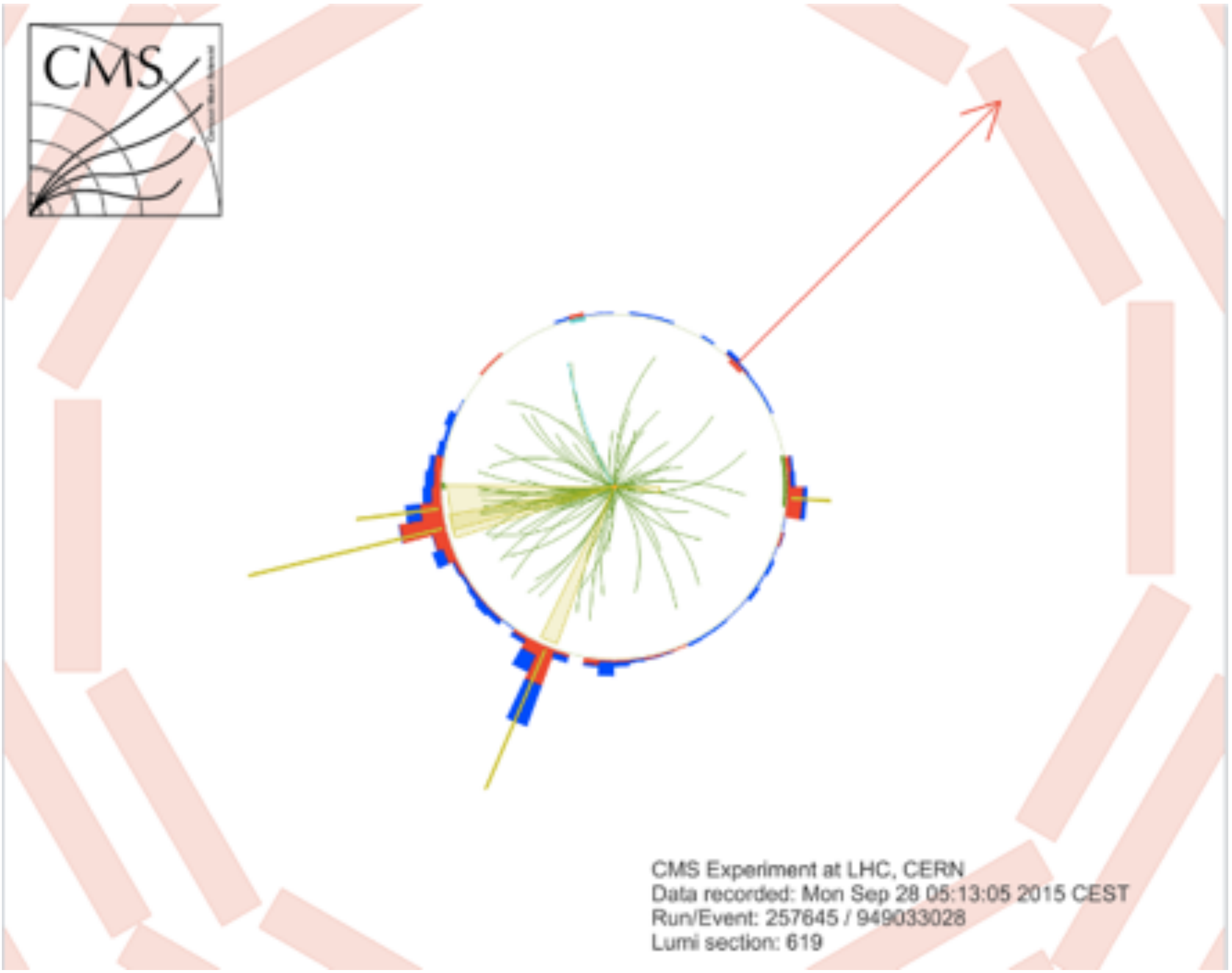
2010-2012 2015-2018 2021-2023 2026-2037

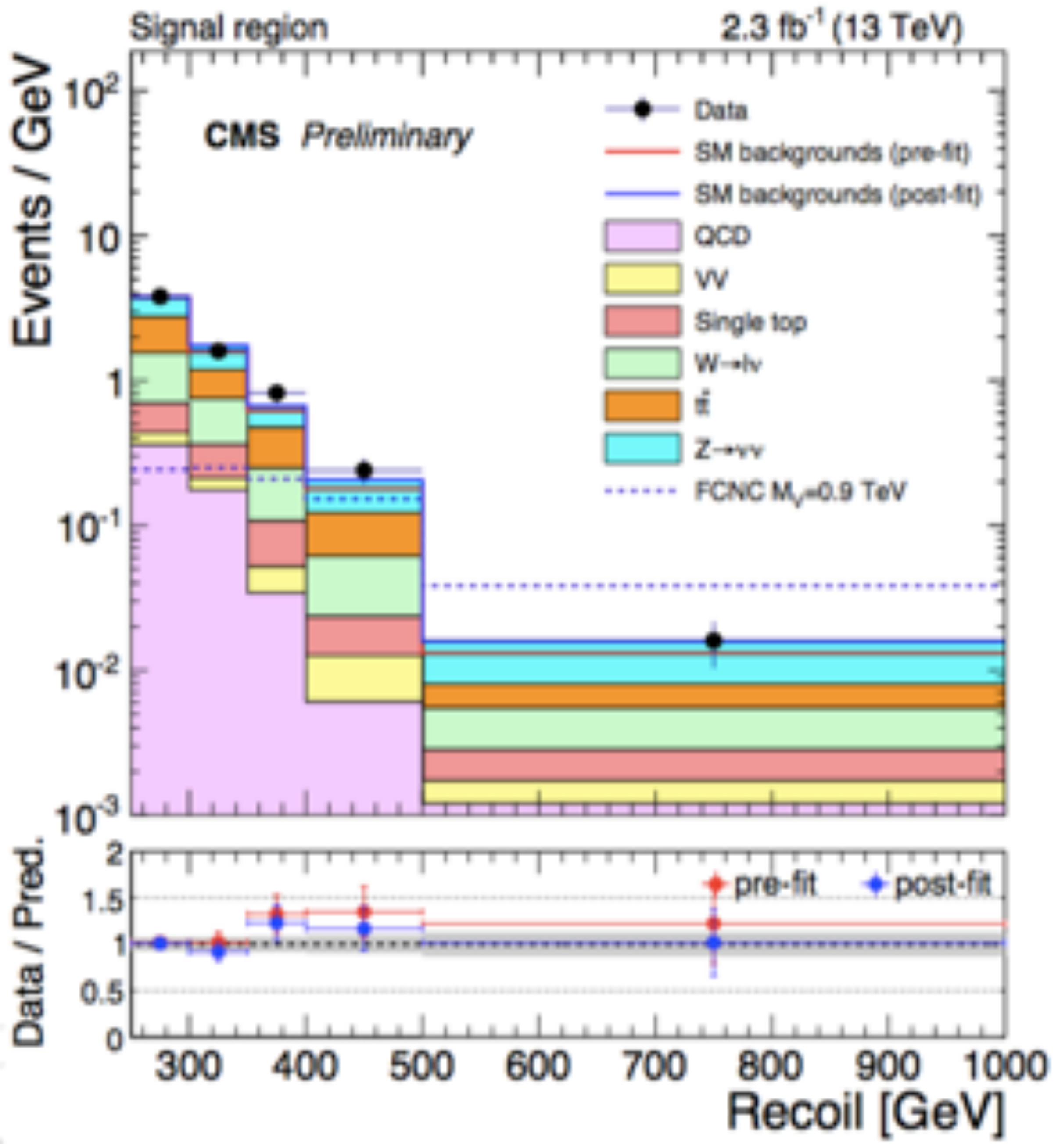
- "Simple" extrapolation of data volume for HL-LHC
  - Extract physics results requires to handle/analyze a lot more data!
- Are industry technologies suitable candidates for user analysis?

Input for the plot: Technical Proposal for the Phase-II Upgrade of the CMS Detector (https://cds.cern.ch/record/2020886)
Main assumption: derived data x8 of RAW data
Use 200 PU events scenario for HL-LHC

Physics use case: Search for Dark Matter

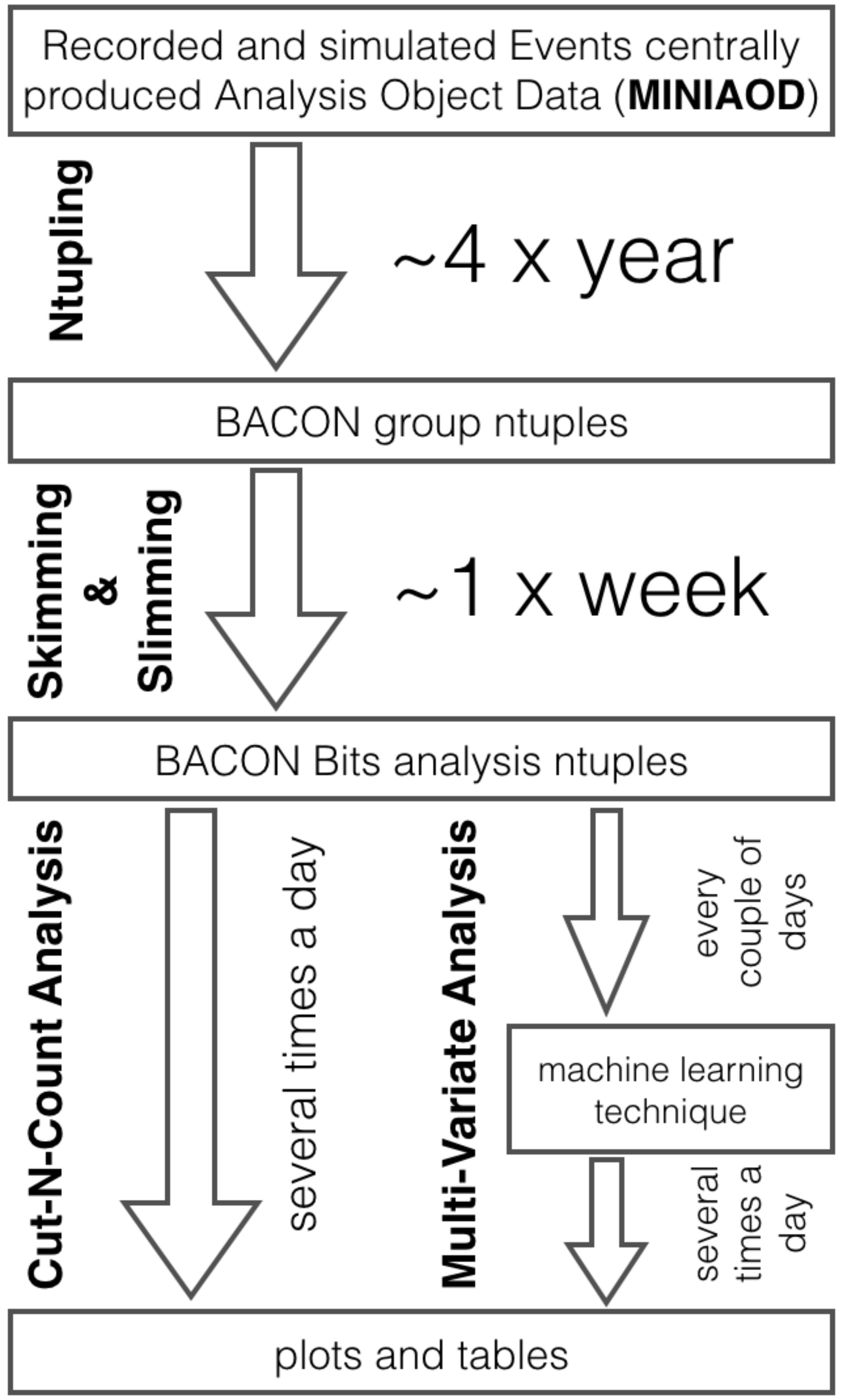






- If it exists, Dark Matter would be produced in association with visible particles.
  - Dark Matter particle(s) would propagate through the detector undetected while visible particles would leave signals in the CMS detector.
- The signature we search for in Dark Matter production at CMS is an energy imbalance, or “missing transverse energy” associated with detectable particles.
  - This signature is commonly referred to as “monoX” where “X” can be a light quark or gluon, a vector boson, or a heavy quark such as a bottom or top quark.
- We focus our search on the “monoTop” signature, where the detectable particle is a top quark

Analysis in ROOT - A multi-step process

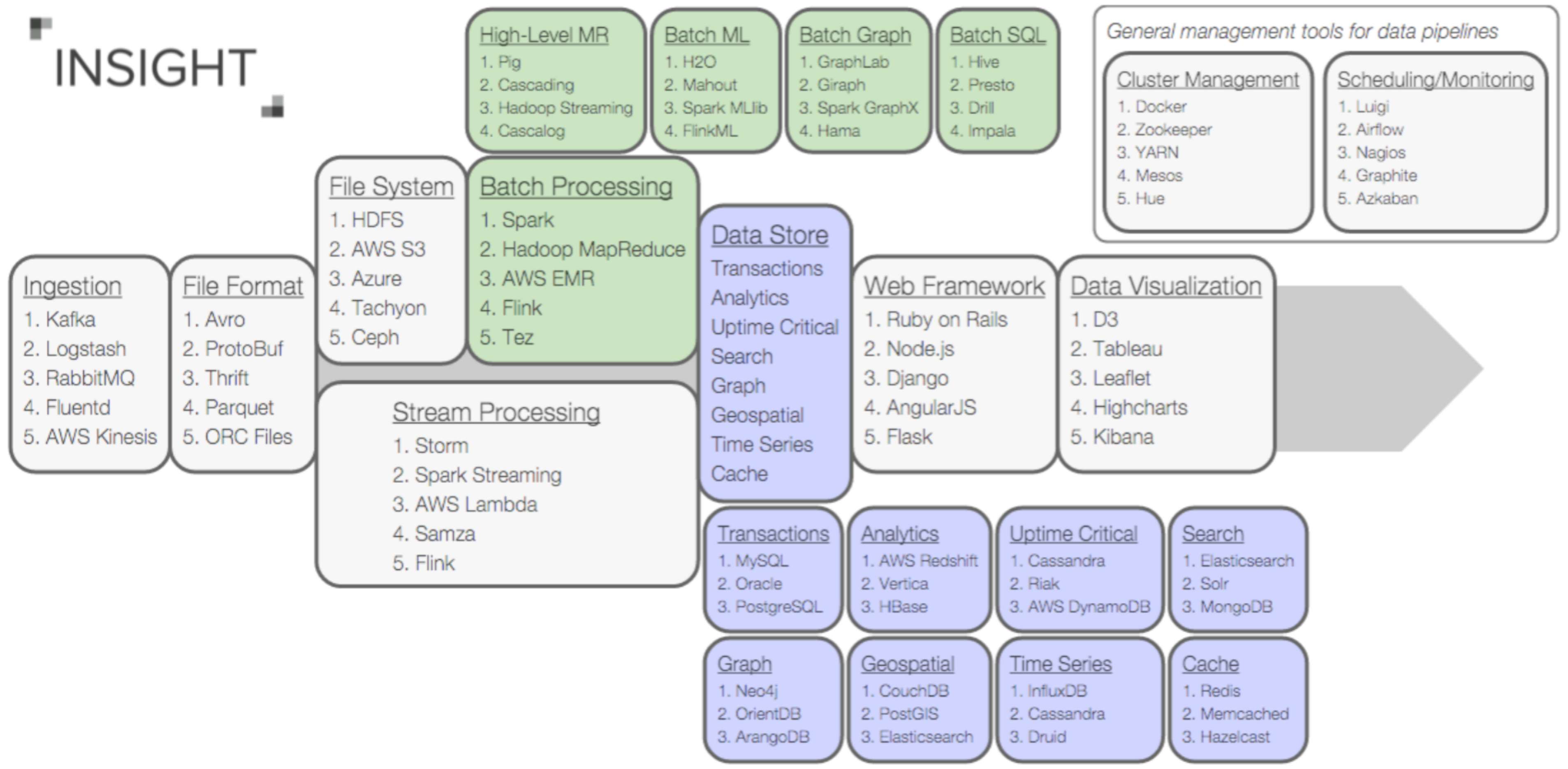


- Interactivity is the key to successful analysis: “Search for the needle in the haystack”
  - Select events, calculate new properties, train neutral nets, etc.
- Collaborations are big, hundreds of physicists are accessing the data
- Current Analysis Workflow
  - Touches only a subset of the total data volume, but subset varies from analysis to analysis
  - Complicated multi-step workflow because dataset is too large for interactive analysis
  - Can take weeks using GRID resources and local batch systems
  - Not all time spent is actual CPU, a lot of time is bookkeeping, resubmission of failed jobs, etc.
- Input:
  - Centrally produced output of reconstruction software, reduced content optimized for analysis
    - Too big for interactive analysis
- Ntupling:
  - **Convert** into format suited for interactive analysis
    - Still too big for interactive analysis
- Skimming & Slimming:
  - Reduce number of events and information content
    - Analysts can explore data and simulation interactively

Big Data

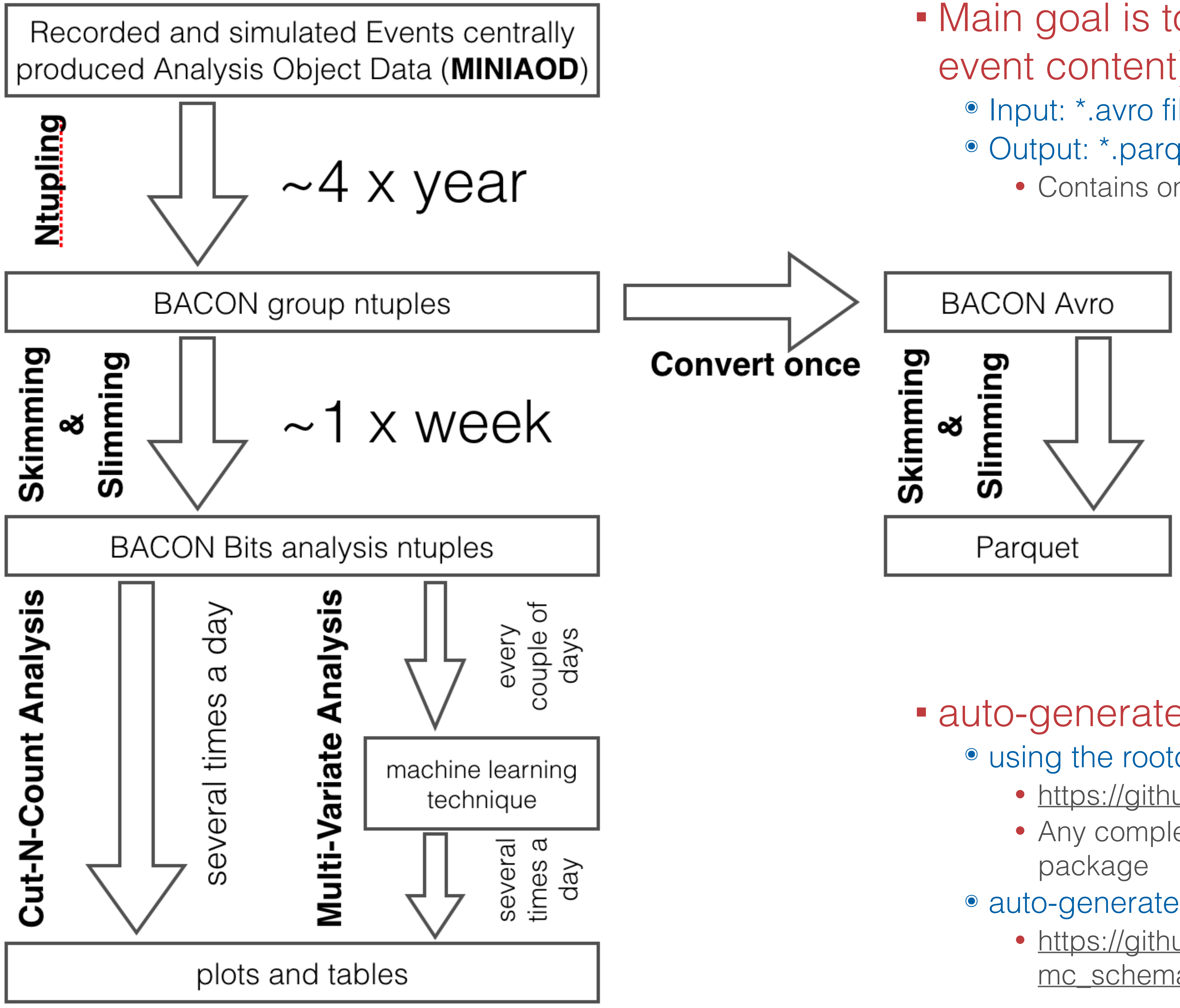
- New toolkits and systems collectively called “Big Data” technologies have emerged to support the analysis of PB and EB datasets in industry.

- Our goals applying these technologies to HEP analysis challenge:
  - Reduce time-to-physics
  - Educate our graduate students and post docs to use industry-based technologies
    - Improves chances on the job market outside academia
    - Increases the attractiveness of our field
  - Use tools developed in larger communities reaching outside of our field



- We want to **use an active LHC Run 2 analysis**, searching for dark matter with the CMS detector, as a **testbed for “Big Data” technologies**
  - Starting point: **Apache Spark**

Spark Workflow



Scala code on Spark

- Main goal is to skim (reduce number of events) and slim (reduce event content).
  - Input: *.avro files (equivalent to big group ntuples)
  - Output: *.parquet files (small size ~1GB) -> useful for analysis:
    - Contains only the information needed i.e. SparkWorkflow performs the main analysis

- auto-generated from the bacon ROOT files:
  - using the rootconverter package:
    - https://github.com/diana-hep/rootconverter
    - Any complex ROOT file can be converted to its corresponding Avro using the same package
  - auto-generated schema for bacon Avro
    - https://github.com/CMSBigDataProject/SparkBaconAnalyzer/blob/master/test/data/mc_schema.avsc

Spark Workflow - Go functional!

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")
```
← Input

```
// First pass (and cache for later)
mcsample.persist()
val mc_sumOfWeights = mcsample.map(_.GenInfo.weight).sum
```
← Sum of Weights for Simulation

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

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

Output contains information of:
- Object (e.g. Muon/Jet)
- Event (e.g. Luminosity) information

```
ntuple.select("mass").show()
...
```
↑ **Physics plots!**

Infrastructure at Princeton

- 10 node SGI Linux Hadoop
  - Intel Xeon CPU E5-2680 v2 @ 2.80GHz CPU processors, 256 GB RAM
  - All servers mounted in one rack and interconnected using a 10 Gigabit Ethernet switch
- Cloudera distribution of Hadoop configured in high-availability mode using two namenodes
  - Spark applications scheduled using YARN
  - External shuffle service inside YARN node manager used to improved stability of memory-intensive jobs with larger number of executor containers
  - Distributed file system (HDFS)
- Converted Bacon Avro stored on the HDFS

Usability tests

We are looking at the "physicist" use case, we are not assuming users to be GRID and HTC experts

Multi-pass workflow beta-tested with two users

Analysis requires sums of event weights as input to analysis code

- ROOT workflow: lxplus/lxbatch cluster at CERN
  - Complicated, uses a script to generate scripts: very complicated and inefficient.
    - Inefficiency could be fixed, but the complexity is a hurdle
  - First pass executed serially
  - Second pass submitted in batch mode (lxbatch)
- Spark workflow: Princeton cluster
  - Two lines of Scala code
  - Spark/Scala caches ("persists") a dataset in the first pass in memory
    - But: Cache maintained manually
  - Second pass over the same dataset mostly or entirely in-memory

Analysis code

- ROOT workflow:
  - Analysis code easy to write and maintain
    - ROOT/C++ is well known in community
- Spark workflow:
  - Scala is a new language
    - Learning curve

Bookkeeping

- ROOT workflow:
  - Scripts designed around specific batch systems (could not be moved easily)
  - Partitioning ("job splitting) handled through sophisticated suite of hand-written shell scripts
    - Relies on physical location of data (i.e. files on EOS at CERN)
- Spark workflow:
  - Very portable (from Princeton system to lxplus in no time)
  - Partitioning can use automatic or custom facilities within Spark
    - example: RDD.repartition(numPartitions: Int)

Performance tests

- Running both the Spark workflow and ROOT workflow on a single lxplus node using one core
  - Input files on local disk: 1 GB ROOT file, 2 GB AVRO file; Caveat: ROOT file is compressed, AVRO is not

	Spark	ROOT
Analysis run without caching	9.4 sec	32.7 sec
Reading from local disk & Computation	*4.3 sec*	26.8 sec
Writing to local disk	5.1 sec	5.9 sec
Analysis run with caching	5.5 sec	
Reading from memory cache & Computation	*0.4 sec*	
Writing to local disk	5.1 sec	

- Conclusion:
  - Comparing the performance of the two is not straight forward, more work needs to go into making the comparison fair
  - Spark is not order of magnitudes slower

Conclusions

- Investigating Big Data technologies to solve the HL-LHC data analysis challenge ➜ Apache Spark as a starting point
  - Fulfills immediately 2 out of 3 goals:
    - Educates our community to use industry-based technologies
    - Uses tools developed in larger communities reaching outside of our field
- In the first pass, we used non-optimized workflows for ROOT and Spark
  - We concentrated on book-keeping and non-optimized performance
- Spark workflow is more user-friendly; ease of use didn't come to a great performance cost (in the limit of the presented comparison)
- Working in parallel on same use case on NERSC resources reading HDF5 files, providing an interesting comparison to presented material
  - Will be presented at the Grace Hopper Conference later this month
- Now we want to dive deeper into the technology and use all its capabilities ➜ Restructure workflow and optimize for respective technology
  - Small-scale test for production of bacon Avro from MINIAOD in CMS software framework environment (CMSSW)
    - https://github.com/nhanvtran/CMSSWToBigData


Fermilab
U.S. DEPARTMENT OF ENERGY
Office of Science