Overview: XENON needs and challenges

Chris Tunnell [Rice U.]

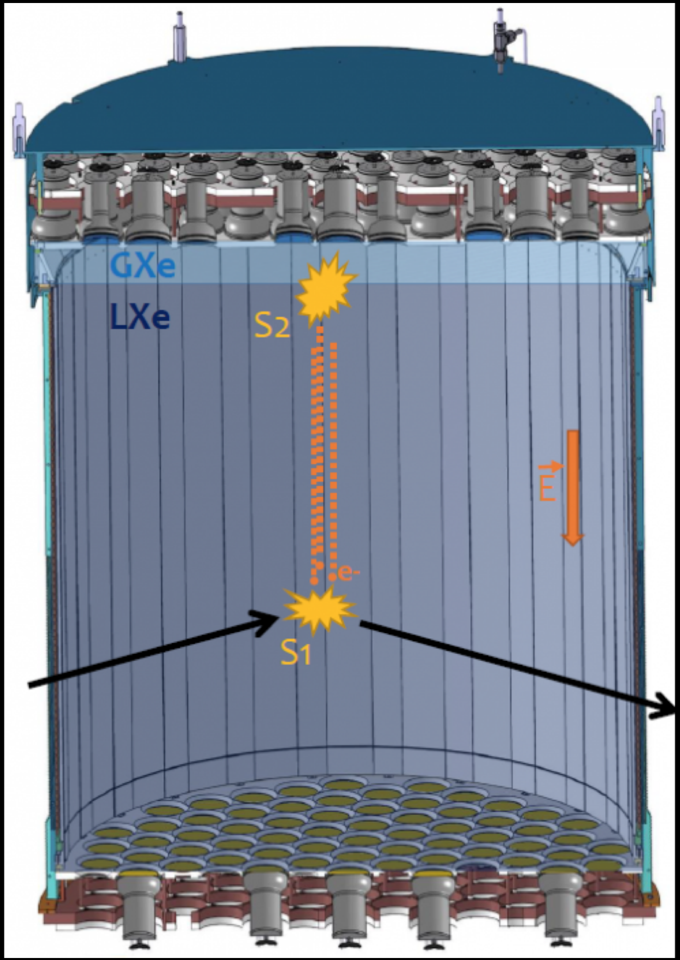
Oct 1, 2019

DANCE 2019

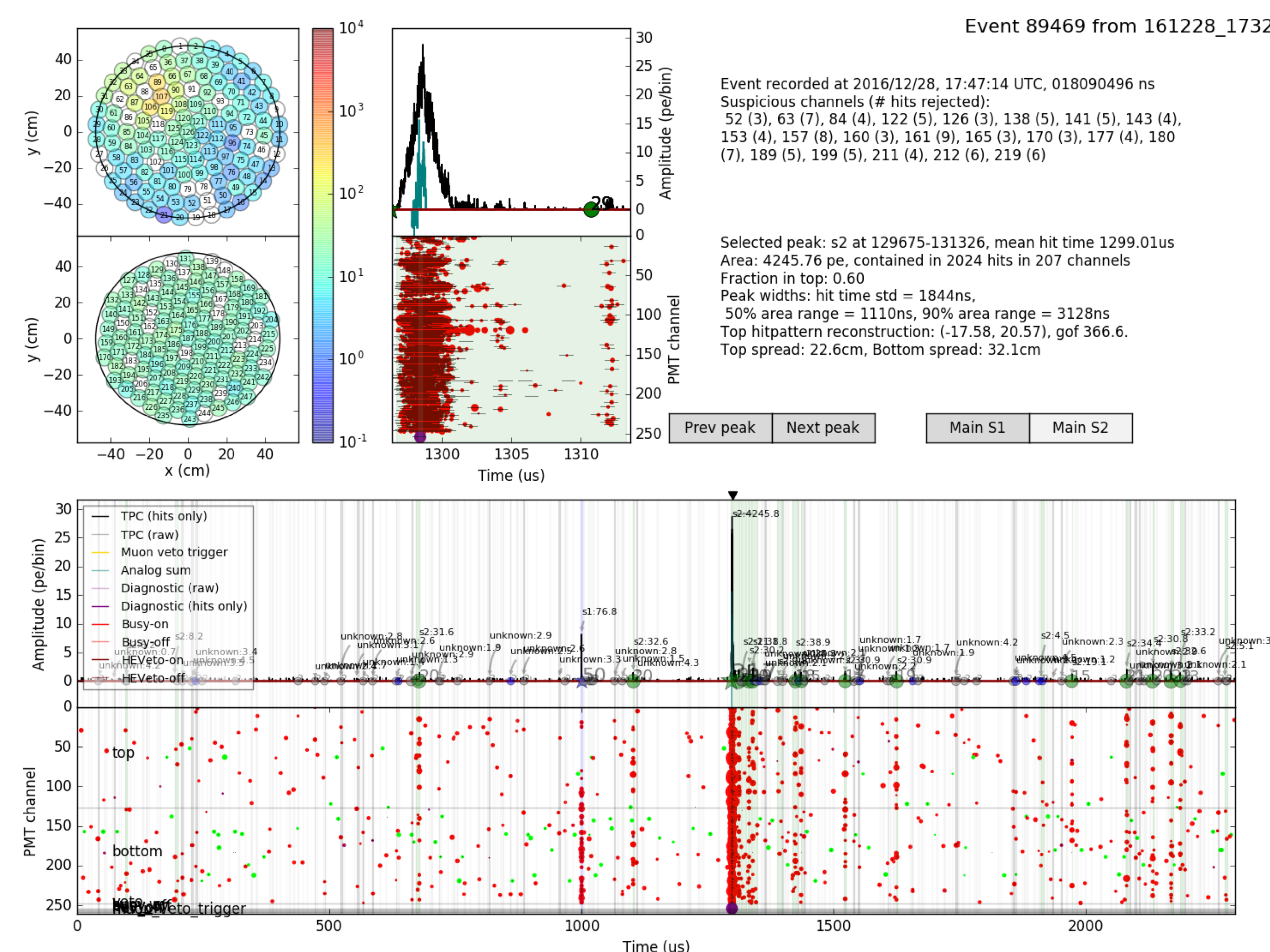
Xe
XENON
Dark Matter Project

What is XENON?

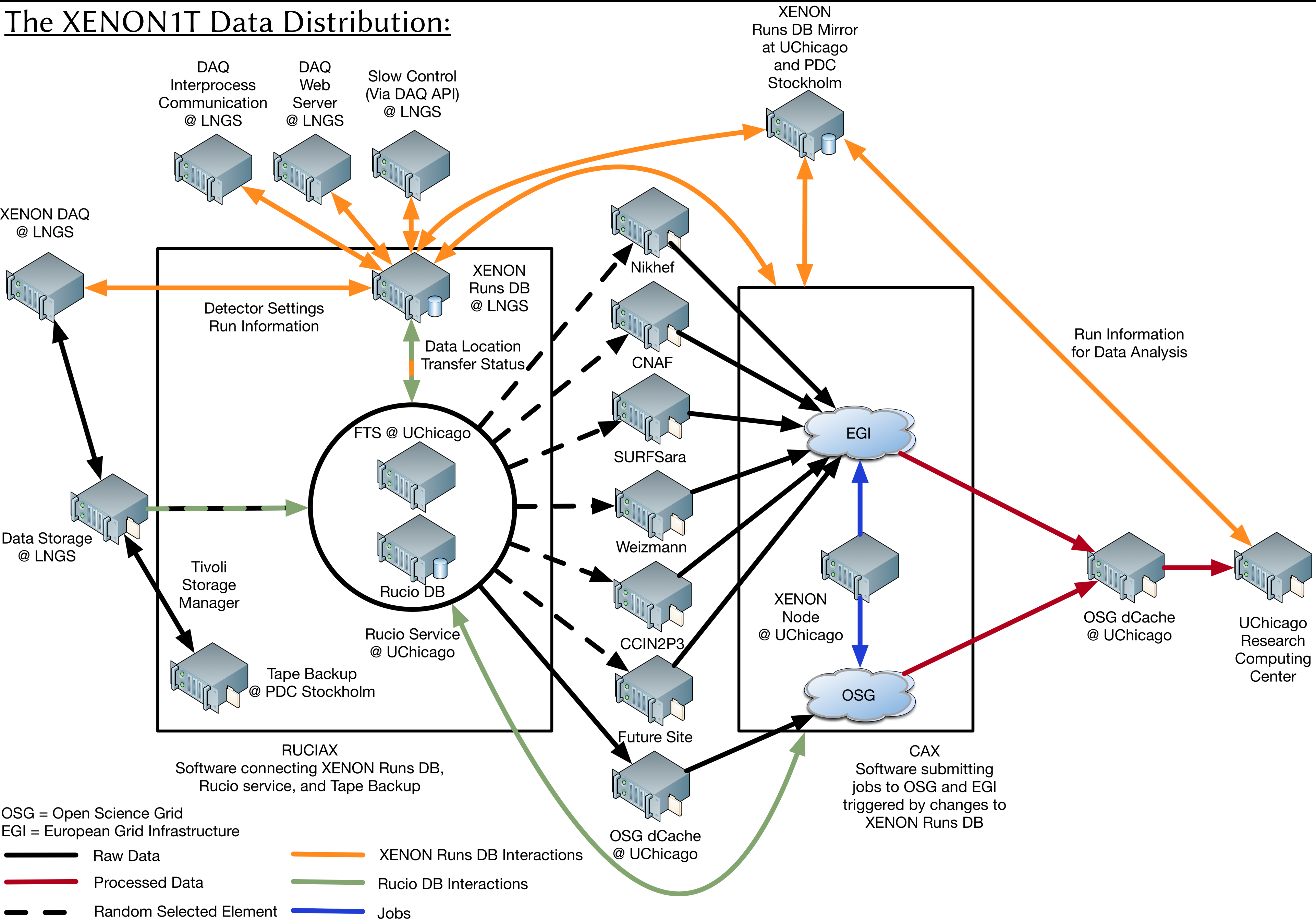
Liquid XENON dark matter detector instrumented with 248 photomultipliers and 10-ns flash ADCs. We make a world-leading new experiment every few years.



arXiv:1805.12562


Event 89469 from 161228_1732
Event recorded at 2016/12/28, 17:47:14 UTC, 018090496 ns
Suspicious channels (# hits rejected):
52 (3), 63 (7), 84 (4), 122 (5), 126 (3), 138 (5), 141 (5), 143 (4),
153 (4), 157 (8), 160 (3), 161 (9), 165 (3), 170 (3), 177 (4), 180
(7), 189 (5), 199 (5), 211 (4), 212 (6), 219 (6)
Selected peak: s2 at 129675-131326, mean hit time 1299.01us
Area: 4245.76 pe, contained in 2024 hits in 207 channels
Fraction in top: 0.60
Peak widths: hit time std = 1844ns,
50% area range = 1110ns, 90% area range = 3128ns
Top hitpattern reconstruction: (-17.58, 20.57), gof 366.6.
Top spread: 22.6cm, Bottom spread: 32.1cm
Prev peak
Next peak
Main S1
Main S2
y (cm)
x (cm)
Amplitude (pe/bin)
PMT channel
Time (us)
TPC (hits only)
TPC (raw)
Muon veto trigger
Analog sum
Diagnostic (raw)
Diagnostic (hits only)
Busy-on
Busy-off
HEVeto-on
HEVeto-off
s2:4245.8
s1:76.8
top
bottom

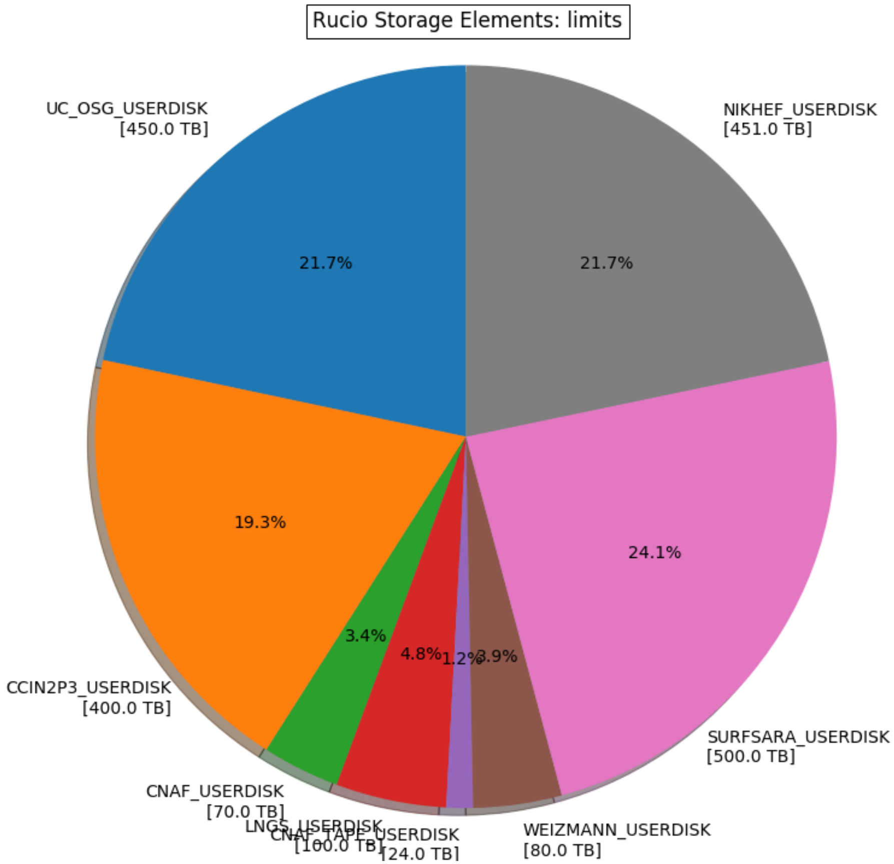
The XENON1T Data Distribution:

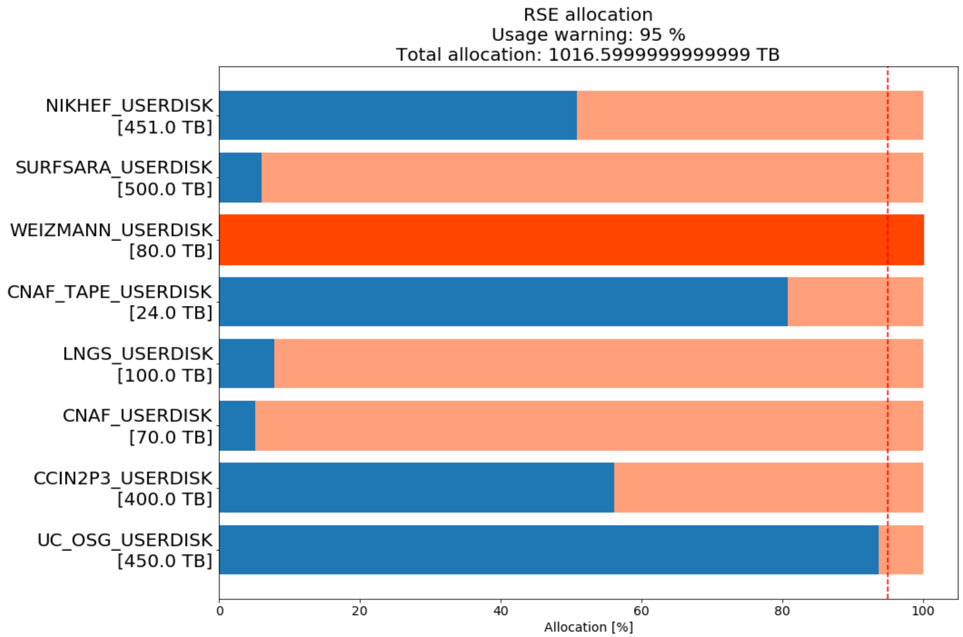


Slide from talk of Boris Bauermeister (U. Stockholm) at 1st Rucio Community Meeting

The XENON1T Disk Allocation and Requirement

- Data have two copies:
  - US: OSG dCache at UChicago (hold only relevant data)
  - Europa: One of several computing centers
- Tape copy in Stockholm Independent from Rucio

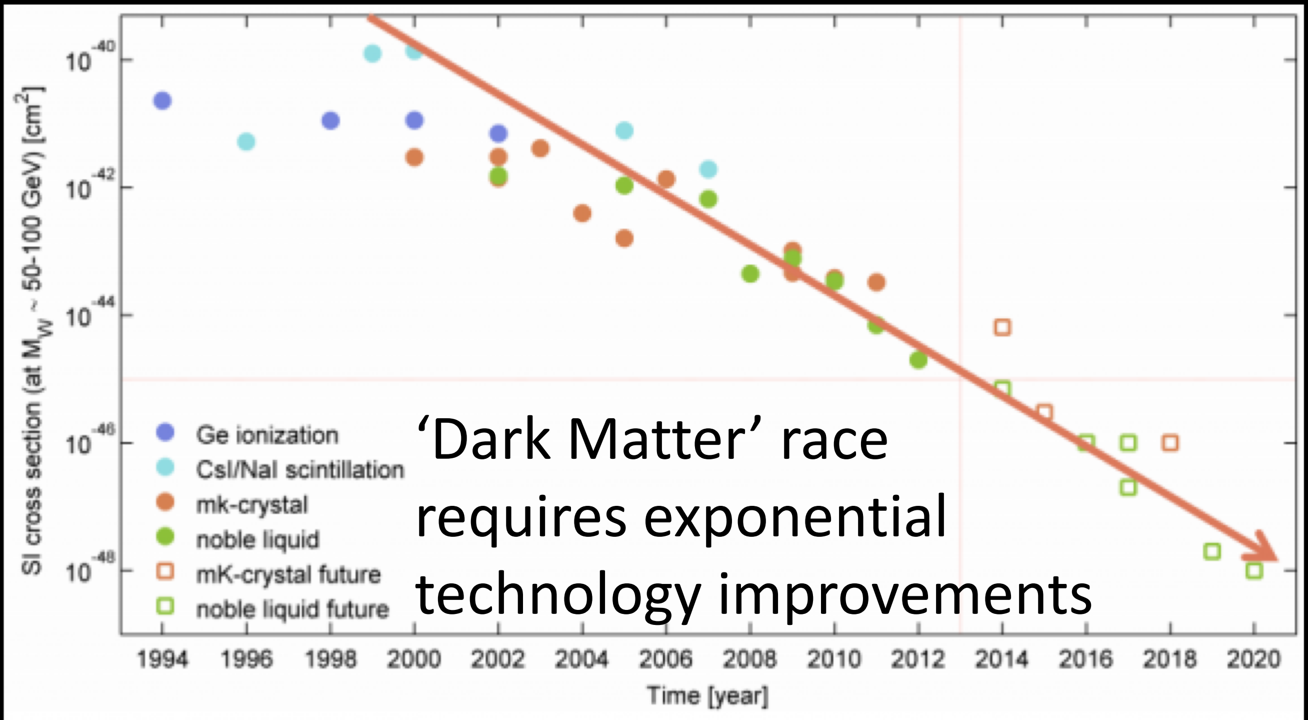




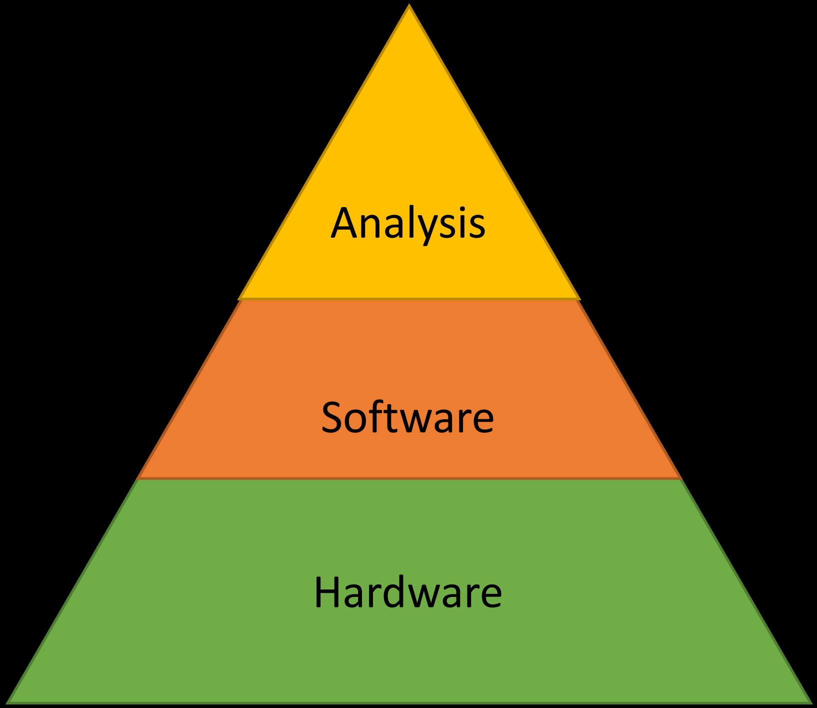
- In total: ~2 PB available
- Distributed worldwide
- Connected to computing centers

Slide from talk of Boris Bauermeister (U. Stockholm) at 1st Rucio Community Meeting

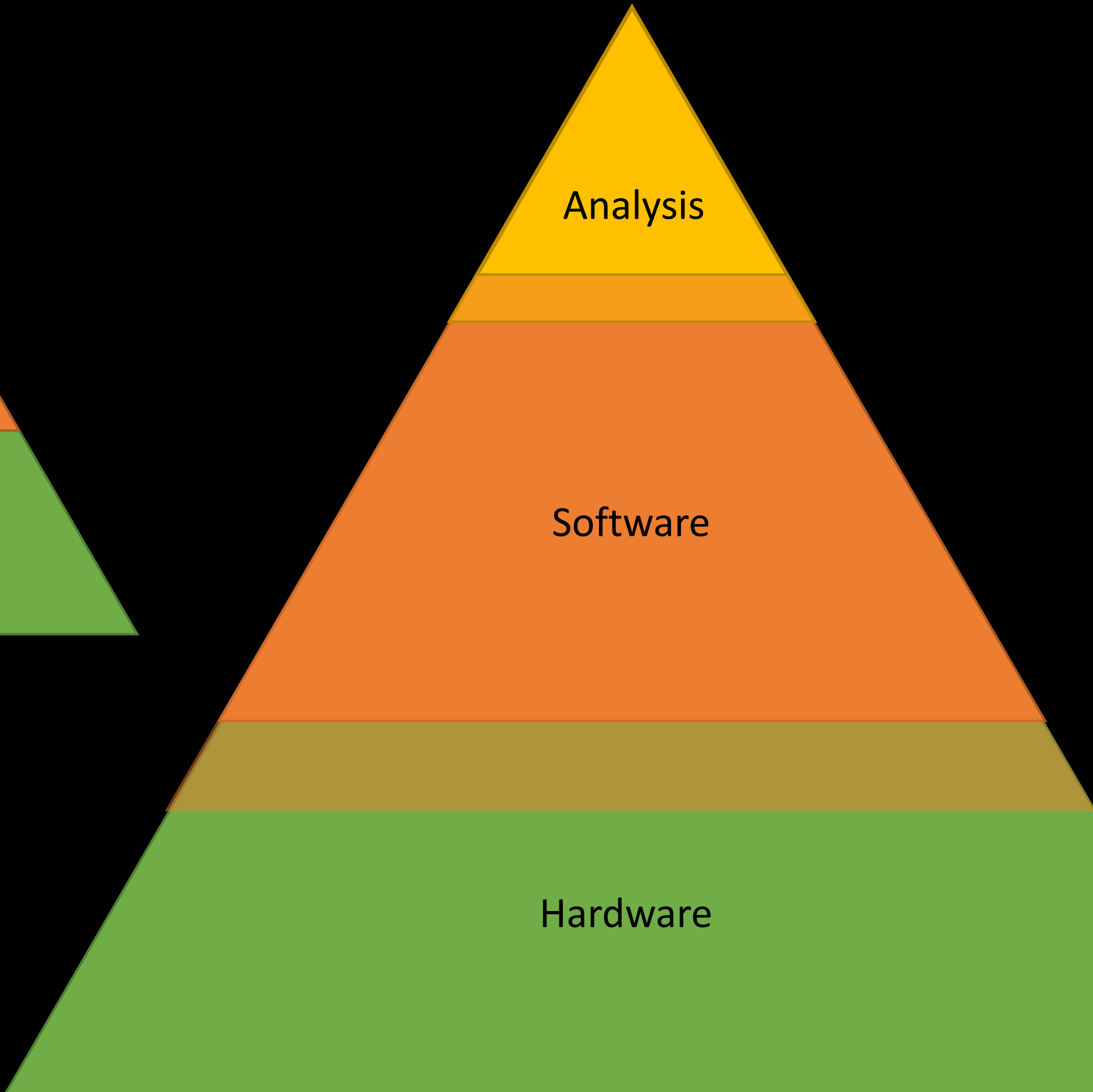
Claim 1: Exponential increase in experimental requirements (incl. computational)



Then



Now



Corollary 2: We became “data intensive” in last decade with requirements per developer also increasing

Transition from informally run unorganized collectives (e.g. XENON10/100) at LNGS to collaboration structure with more division of labor (XENON1T) spreading data around.

How keep up?

Claim 3:
Few inter-collaboration computational R&D efforts, despite need and funding availability

Claim 4:
Our requirements mismatch with other communities, hard to benefit from their prior work

XENON: Using LHC data management tools was (worthwhile) struggle for a while to adapt these tools to us

LZ: NERSC is for high performance instead of high throughput computing, so refactoring overhead

DarkSide: User management

Corollary 4:
LHC experiments have support timescales and legacy requirements irrelevant for us

- We build experiments every few years instead of decades
- Fewer developers, fewer users
- Smaller code base

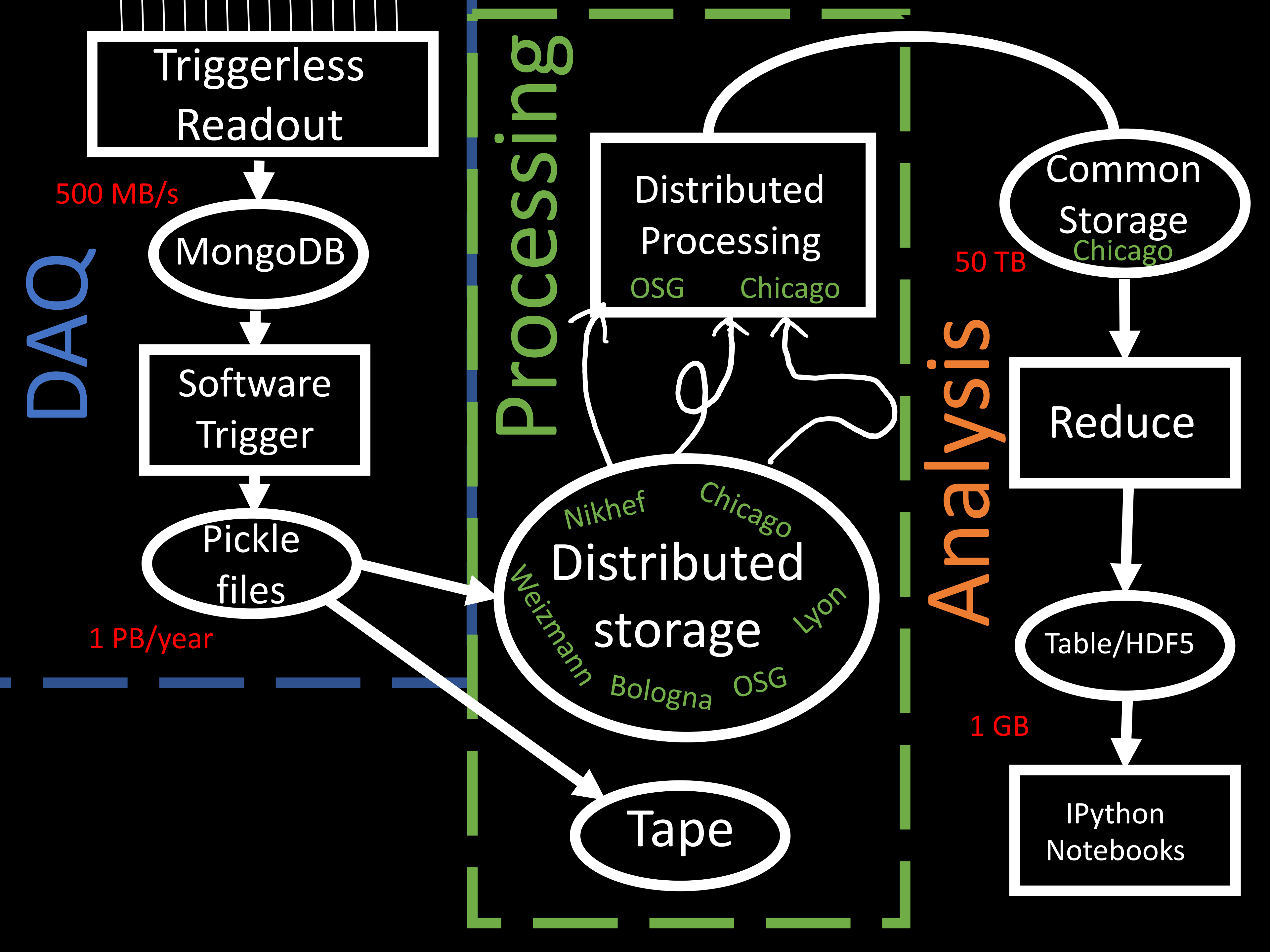
Claim 5:
Around LHC commissioning, technology companies surpassed HEP in Big Data

- Can we follow the "LSST" model of organically using new things to lower manpower requirements and increase science output?
- Can we forget ROOT and join another community that is better supported and transferable?

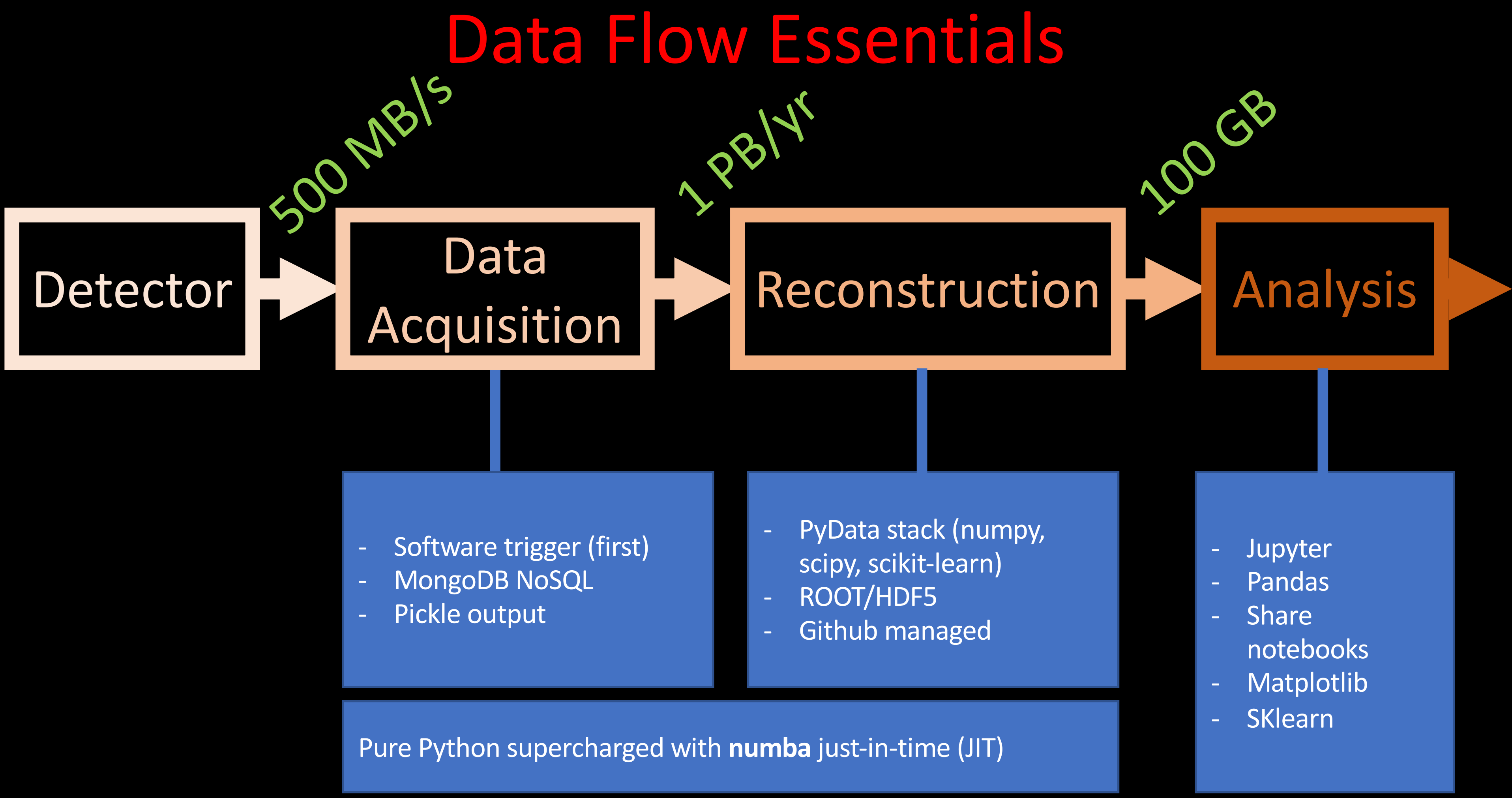
Claim 6:
All general frameworks s*ck, so better to modularize to limit scope where useful

e.g. matplotlib plotting, uproot/HDF5 I/O, nestpy
NEST microphysics, g4py

What does this mean for software development?



DAQ
Triggerless Readout
500 MB/s
MongoDB
Software Trigger
Pickle files
1 PB/year
Processing
Distributed Processing
OSG
Chicago
Distributed storage
Nikhef
Chicago
Weizmann
Lyon
Bologna
OSG
Tape
Analysis
Common Storage
Chicago
50 TB
Reduce
Table/HDF5
1 GB
IPython Notebooks


Data Flow Essentials
500 MB/s
1 PB/yr
100 GB
Detector
Data Acquisition
Reconstruction
Analysis
- Software trigger (first)
- MongoDB NoSQL
- Pickle output
- PyData stack (numpy, scipy, scikit-learn)
- ROOT/HDF5
- Github managed
- Jupyter
- Pandas
- Share notebooks
- Matplotlib
- SKlearn
Pure Python supercharged with **numba** just-in-time (JIT)

Challenges and Needs

- Challenge: “Crank up the gain” necessary for dark matter program
  - Effect: 1) 500 MB/s streaming rate to 2) PB/yr storage rate
  - New challenge 1): How to build fast upgradable streaming system?
    - Subchallenge: Physics education fails to teach basic skills in programming, Data Science, or transferable skills… so we have to teach this as best we can
    - Subchallenge: Sit between HEP and High Tech, where they don’t play together
      - Subsubchallenge: Building advanced system with few people requires being frantically technologically aggressive
  - New challenge 2): How to handle PB of data with minimal manpower?
    - Subchallenge: NSF does not have national labs, so highly distributed
    - Subchallenge: Efforts to use prior investment generally has burdensome overhead for our scale experiment
    - Subchallenge: Retaining people with these skills difficult
      - Subsubchallenge: No clear training path for undergraduates, graduates, postdocs, staff
      - Subsubchallenge: Also no clear career trajectory
      - Subsubchallenge: Lack of steady operational funds
    - Subchallenge: How do analysts interact with these data without giant software stacks and in an interactive manner (e.g. JupyterHub)?

Numba: Python just-in-time compiler

- Few ‘array-oriented’ compilers though common use case and hardware optimizations exist.
- Wasn’t possible few years ago, **Python faster than your C++ code.**

```
@vectorize
def sinc(x):
    if x==0.0:
        return 1.0
    else:
        return sin(x*pi)/(pi*x)
```

```
; ModuleID = 'sinc_mod_7b29370'

define double @sinc(double %x) {
Entry:
  %0 = fcmp oeq double %x, 0.000000e+00
  br i1 %0, label %BLOCK_12, label %BLOCK_16

BLOCK_12:                                         ; preds = %Entry
  ret double 1.000000e+00

BLOCK_16:                                         ; preds = %Entry
  %1 = fmul double %x, 0x400921FB54442D18
  %2 = call double @llvm.sin.f64(double %1)
  %3 = fmul double %x, 0x400921FB54442D18
  %4 = fdiv double %2, %3
  ret double %4

BLOCK_47:                                         ; No predecessors!
  ret double 0.000000e+00
}

declare double @llvm.sin.f64(double) nounwind readonly
```

CONTINUUM ANALYTICS

Backup

- Details of computing scheme in arXiv: 1903.11441
- Code under XENON1T, XENONnT, AxFoundation Github organizations