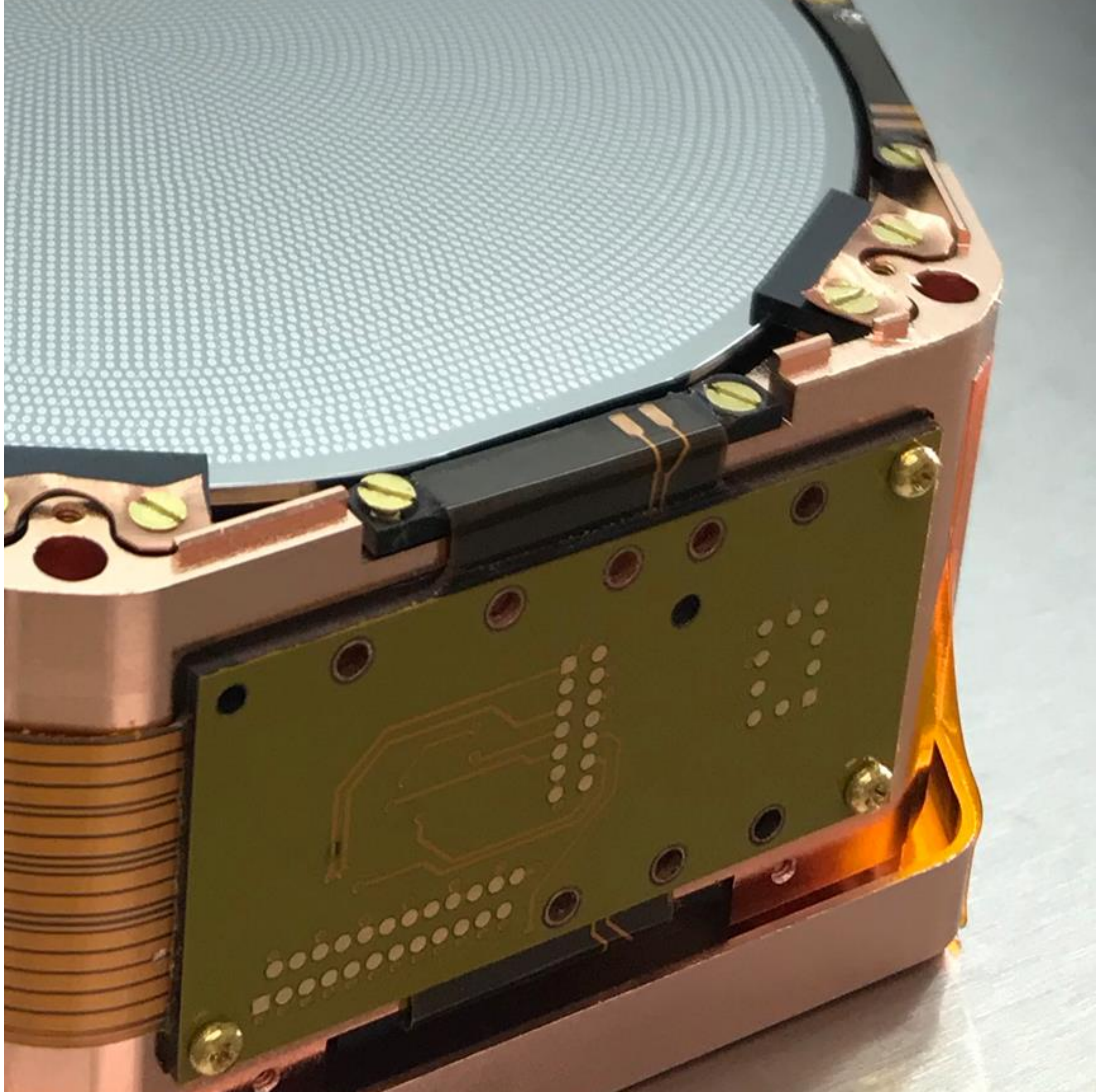
Software Plans and Challenges for SuperCDMS

Ben Loer
Pacific Northwest National Laboratory



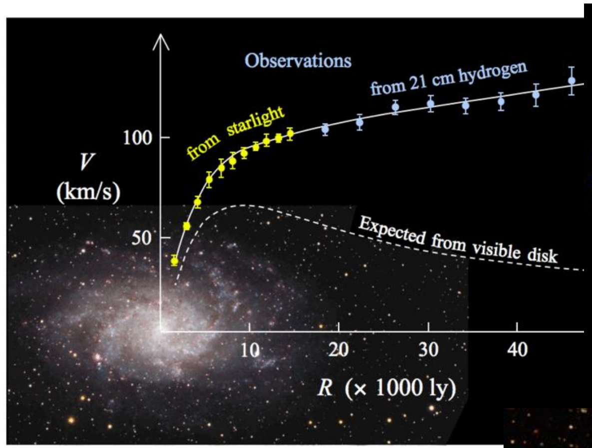
PNNL is operated by Battelle for the U.S. Department of Energy

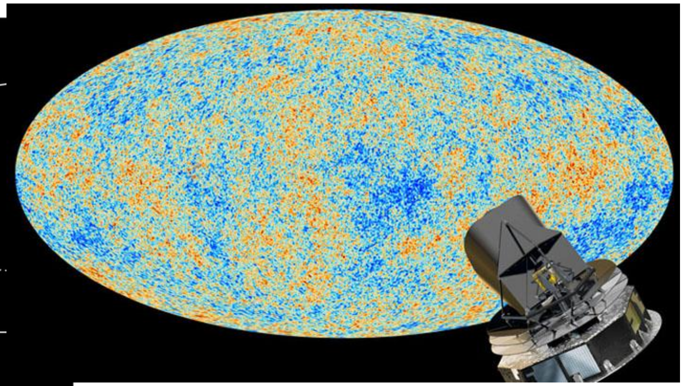
Outline

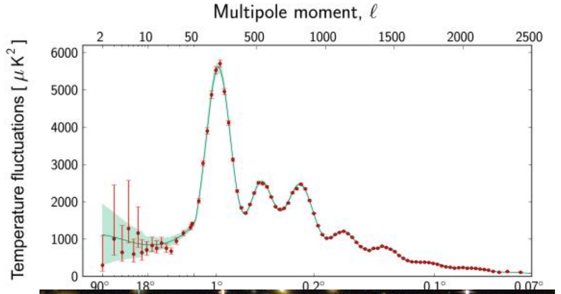
- Brief intro to direct dark matter detection and SuperCDMS technology
- Overview of the SuperCDMS software stack
- Peculiarities of data and analysis
- Current development and needs

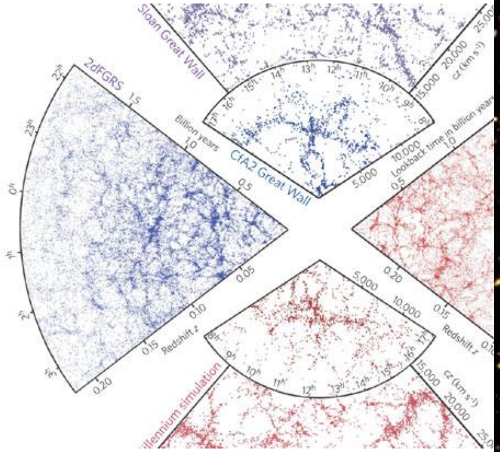
- NB:
  - I am probably misusing lots of common HEP terms
  - “You’re doing that wrong” is totally helpful feedback and encouraged

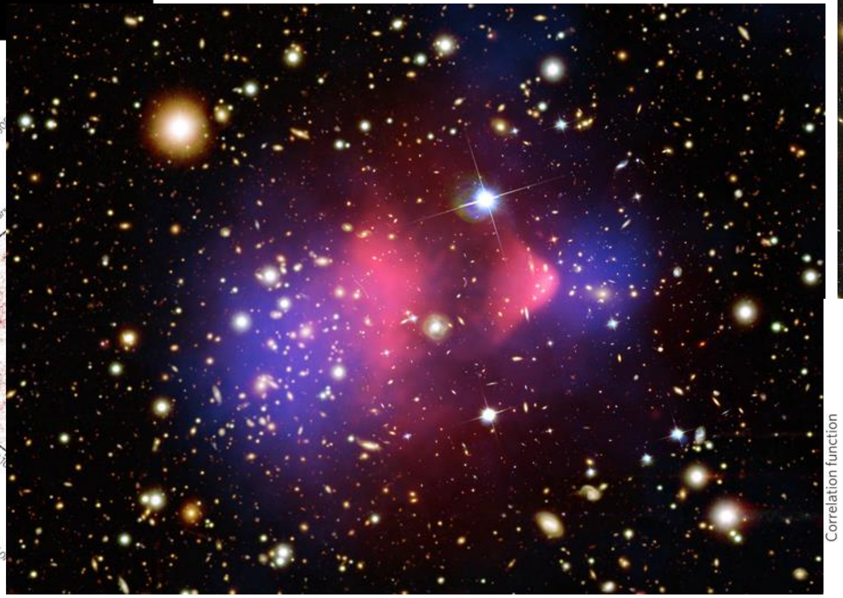
Evidence for dark matter

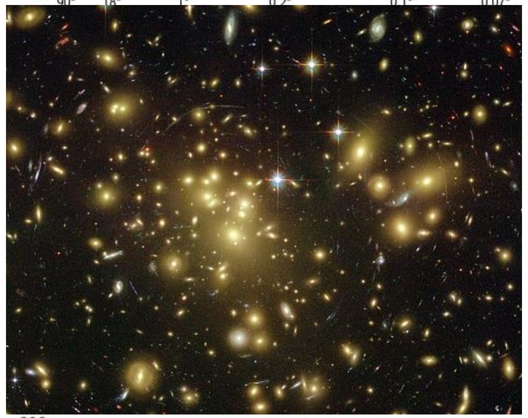


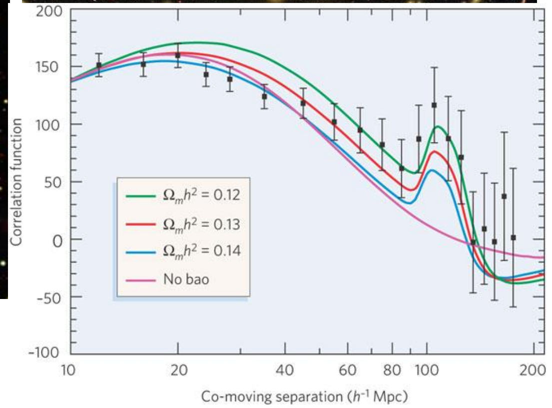








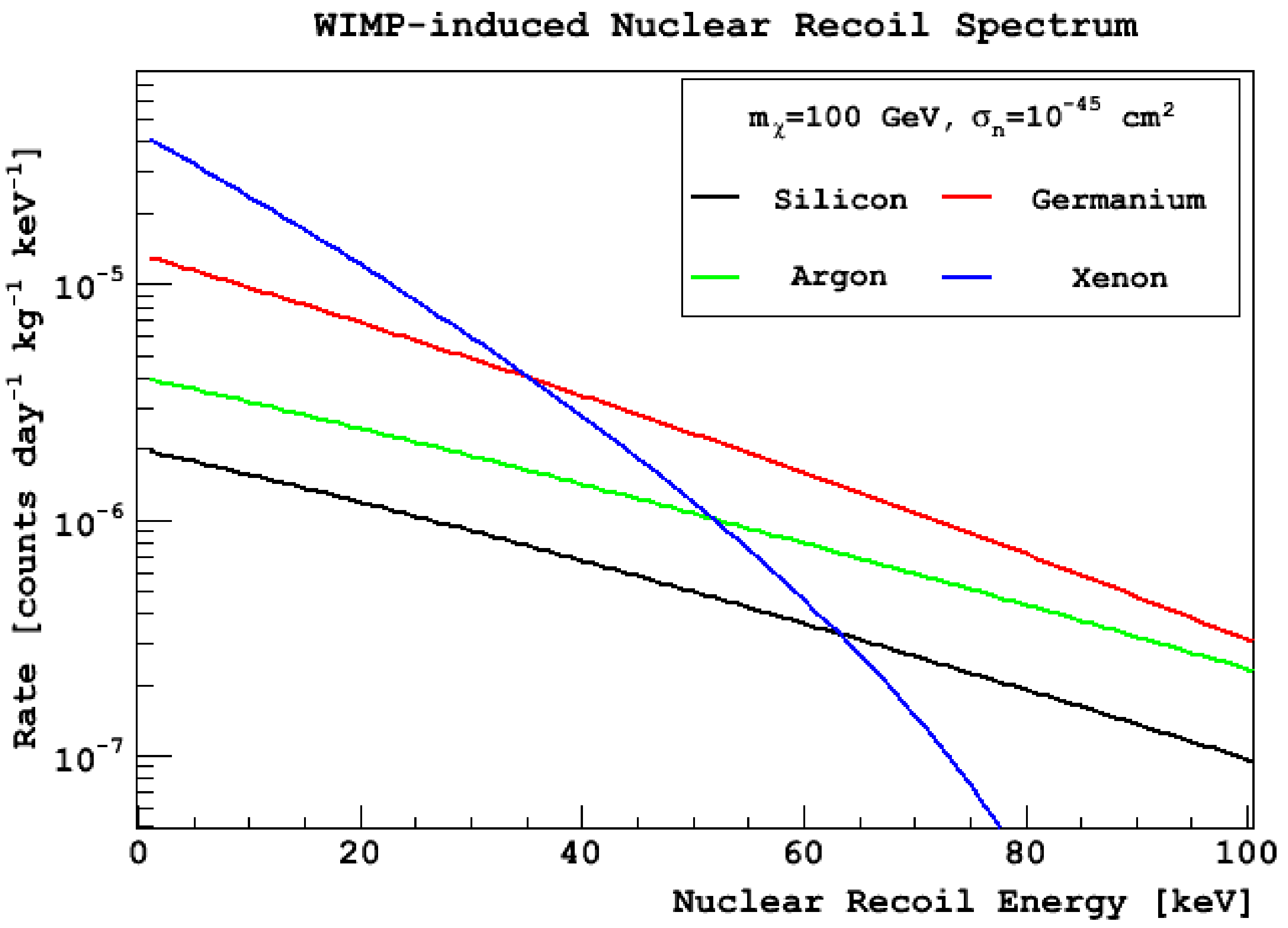




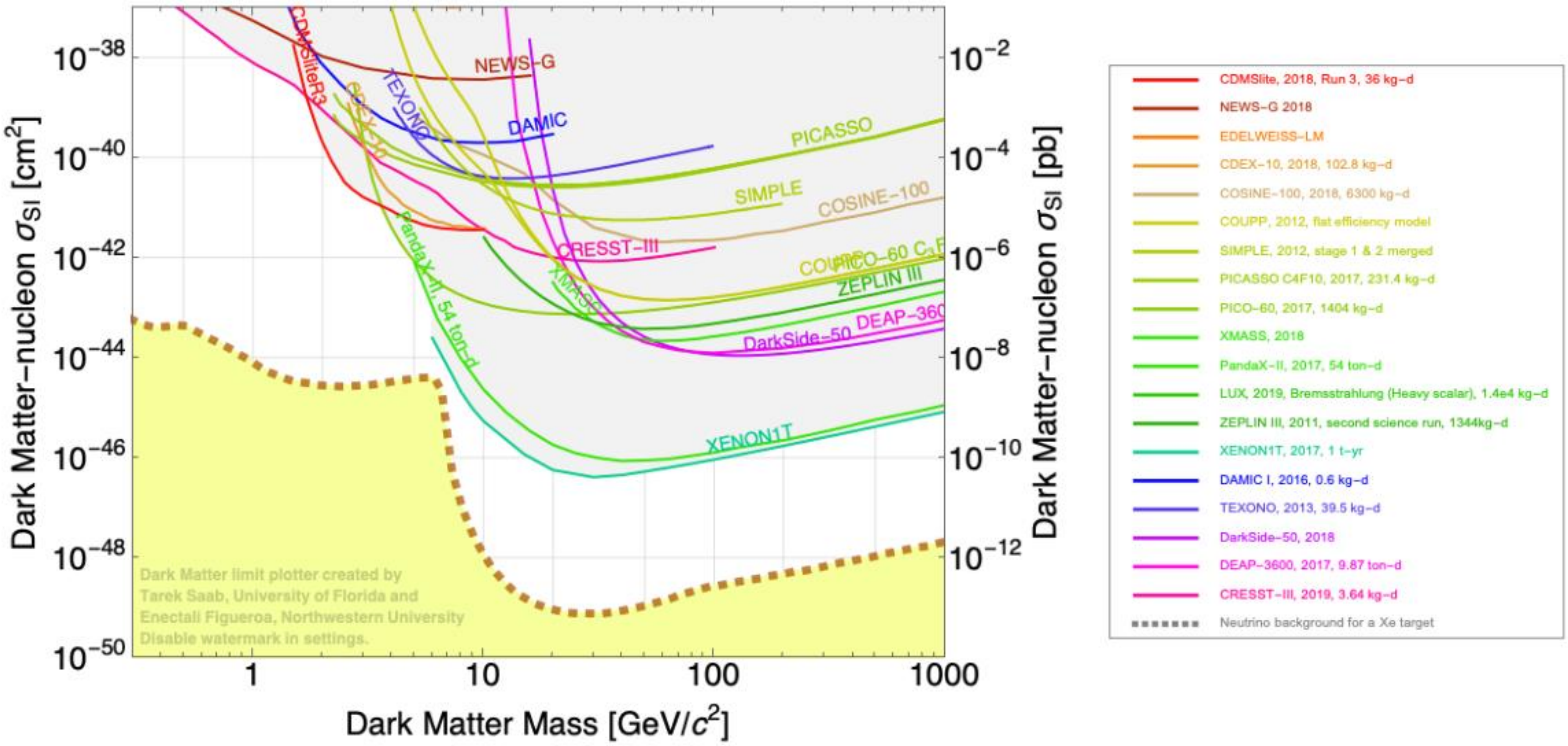
Dark Matter Direct Detection

- Assume dark matter is a gas in thermal equilibrium
  - Velocities follow a Maxwell-Boltzmann distribution with v_{avg} ~ 220 km/s
  - Truncated at galactic escape velocity (~600 km/s)
- Boost into lab frame due to sun's motion relative to galaxy
  - Earth's rotation about the sun introduces a small annual modulation that is usually averaged out
- DM interactions generate nuclear recoils in target
  - A^2 coherent enhancement at low momentum transfer
  - Looks like a neutron (minus multiple scatters and thermal capture)
- Result:
  - exponential-ish spectrum up to O(10) keV depending on DM mass
  - Current limits ~1 event /100 kg*years

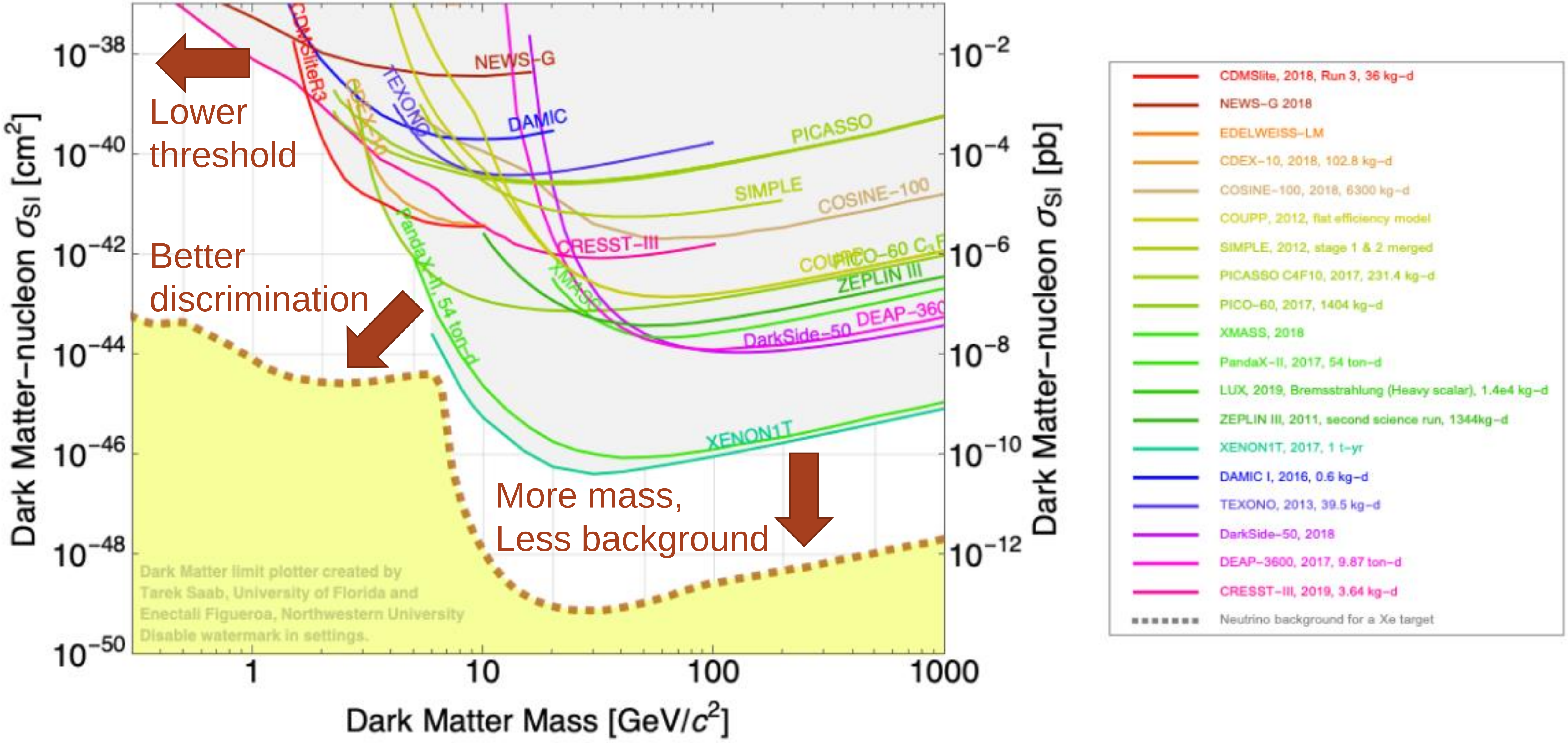
Nuclear recoil spectrum



Direct Detection Landscape



Direct Detection Landscape

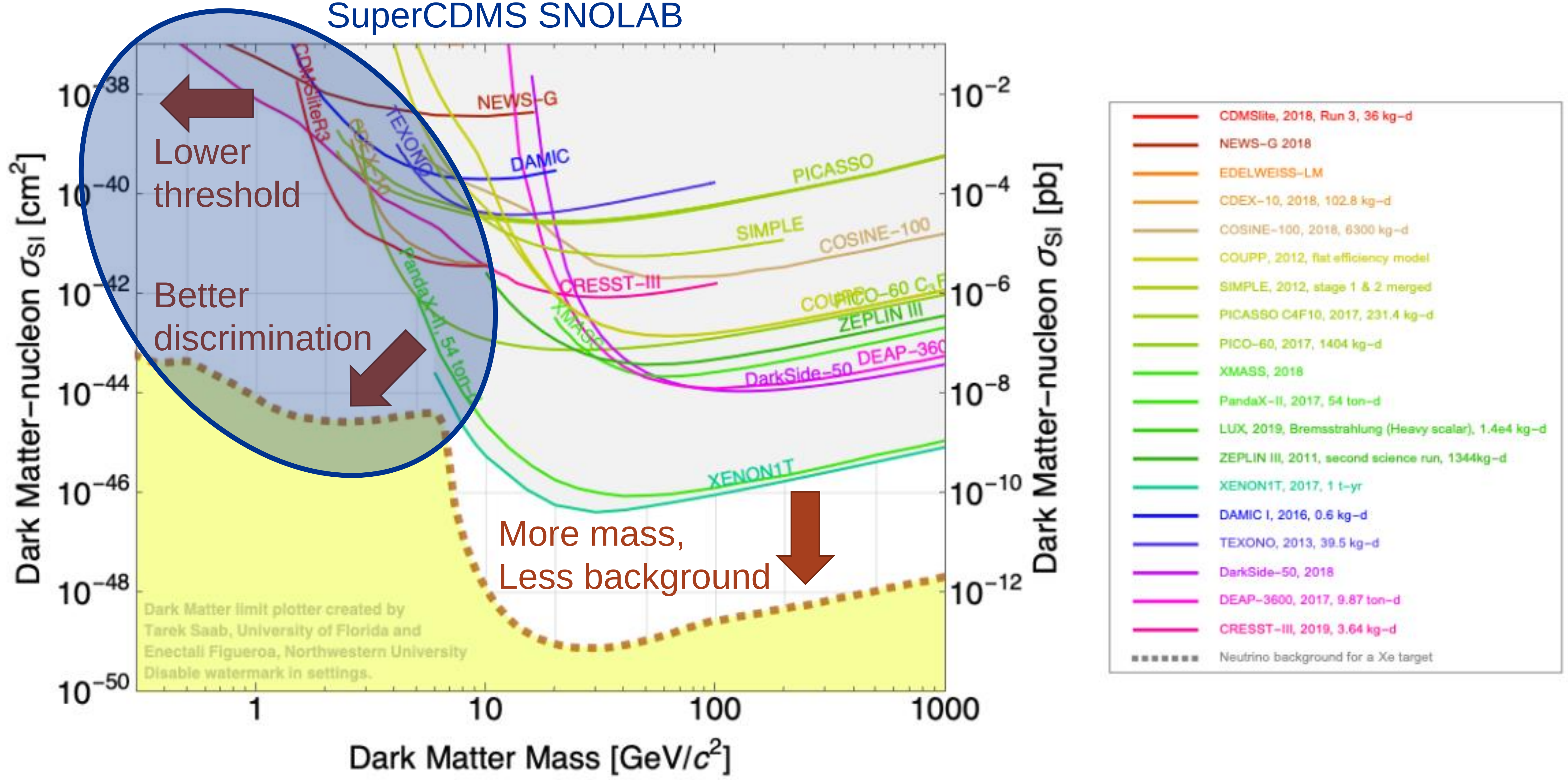


Pacific Northwest
NATIONAL LABORATORY

Super CDM

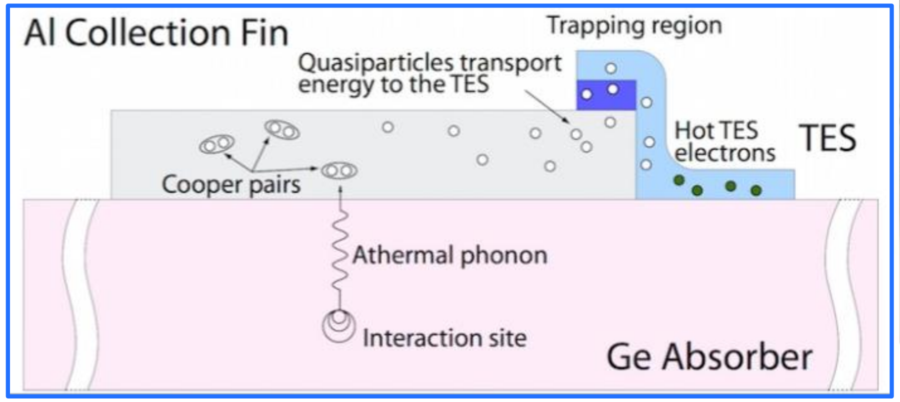
Direct Detection Landscape

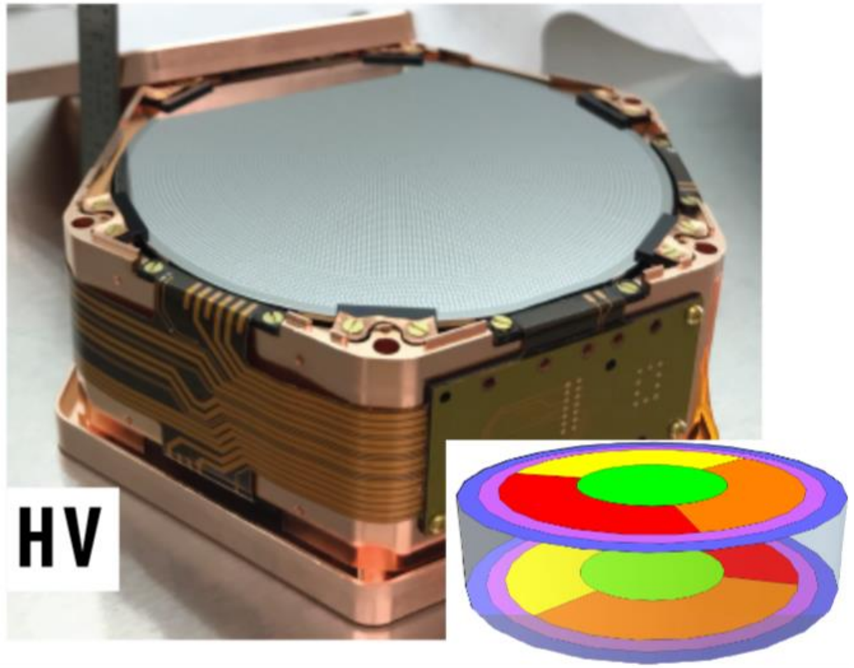
SuperCDMS SNOLAB

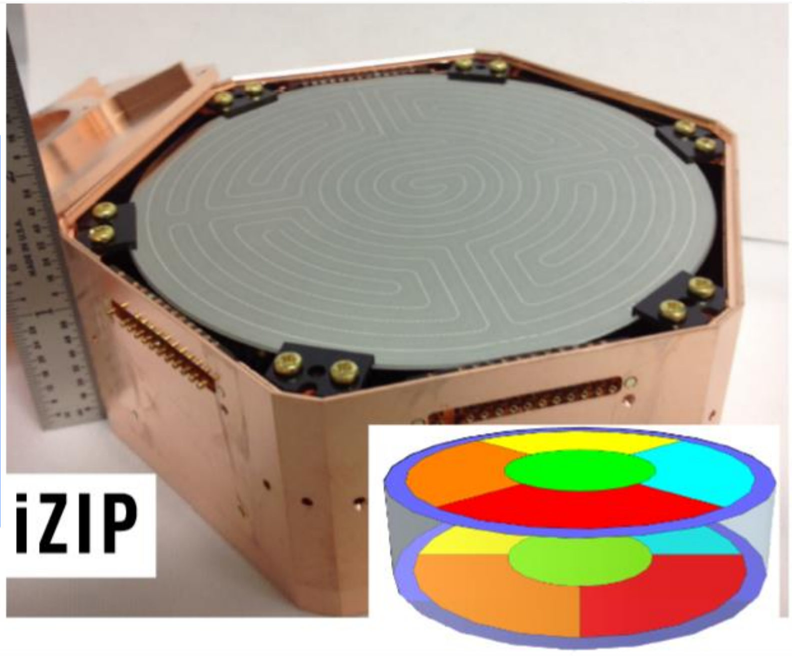


SuperCDMS Detectors

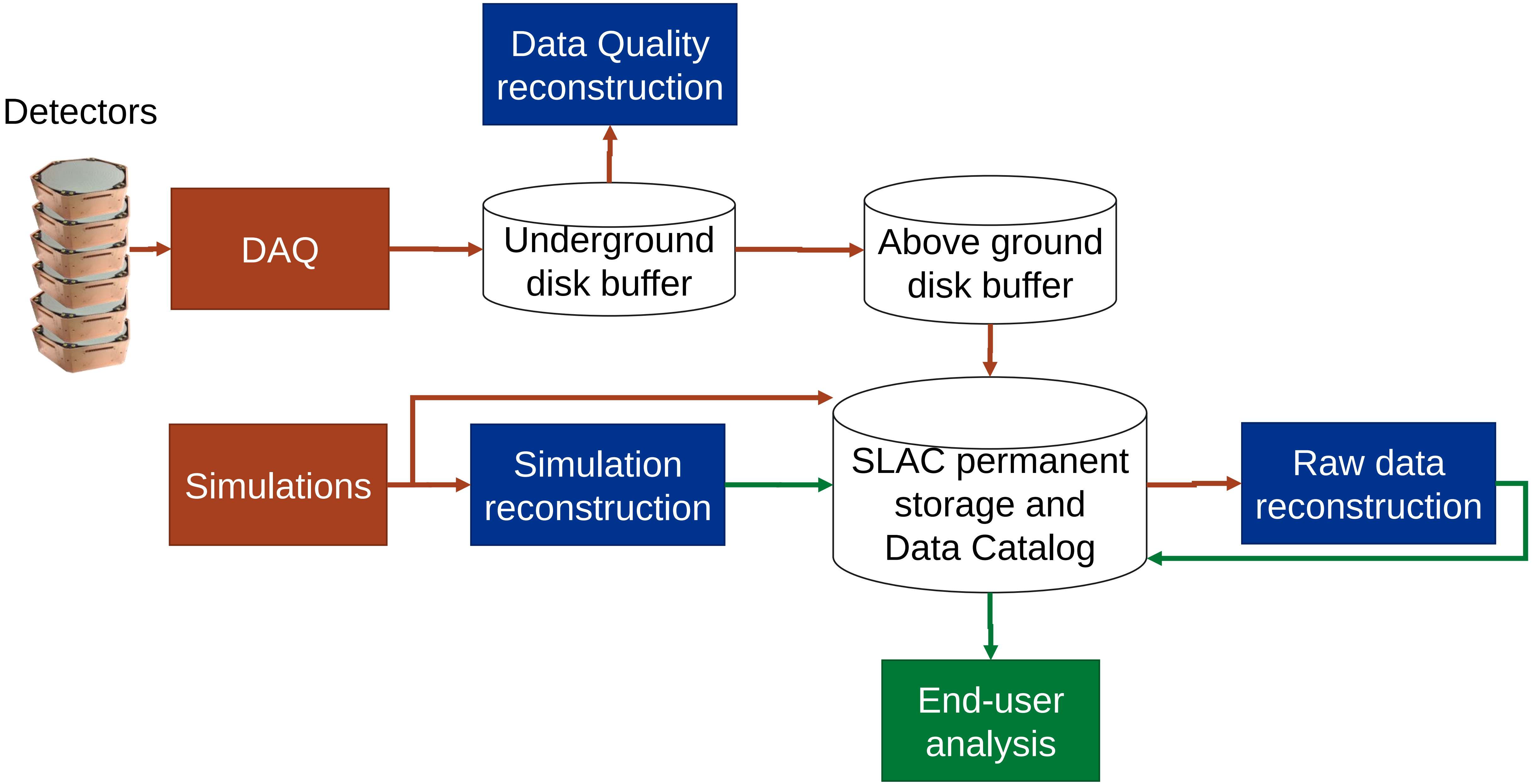
- 100x30 mm pure Ge or Si crystals
- Athermal phonons collected into superconducting transition edge sensors (TES), read by SQUID amplifiers
- **Extremely sensitive to noise and environment**
- HV detectors convert free charge to additional phonons with high gain and low noise
  -> **<1 e- threshold**
- iZIP detectors read ionization
  -> q/E discrimination







SuperCDMS Software Stack



Pacific Northwest NATIONAL LABORATORY

SuperCDMS Collaboration

- ~25 institutions, ~125 members
- ~5 FTEs total for all software infrastructure, split between
  - DAQ
  - Simulations
  - Data processing and analysis environment

Data Acquisition

- System built on MIDAS DAQ library from TRIUMF
  - Used by DEAP3600, T2K, g-2
  - Independent processes communicate through shared memory buffers and network RPC calls
  - State and configuration handled by central Online Database (ODB)
- Detector channels read by Detector Control and Readout Cards (DCRCs)
  - Custom board with firmware design similar to mu2e SiPM boards
  - Signals digitized into 6s long circular buffer: sets max trigger + readout time
  - L1 triggers generated by digital FIR filter
- L2 trigger process collects L1 triggers from DCRCs, requests event readout
- Event builder collects and orders traces for writing to disk

Data volumes

- 24 detectors x 12 (16) channels for HV (iZIP) detectors
- ~0.14 MB (uncompressed) per detector per trigger
- Highest data rate during calibration, ~5 Hz per detector ->O(10) MB/s
- O(100) TB/year raw data, similar amount of simulation data
- O(10) TB/year ntuples
  - Just barely above threshold where a single server is impractical

Simulations

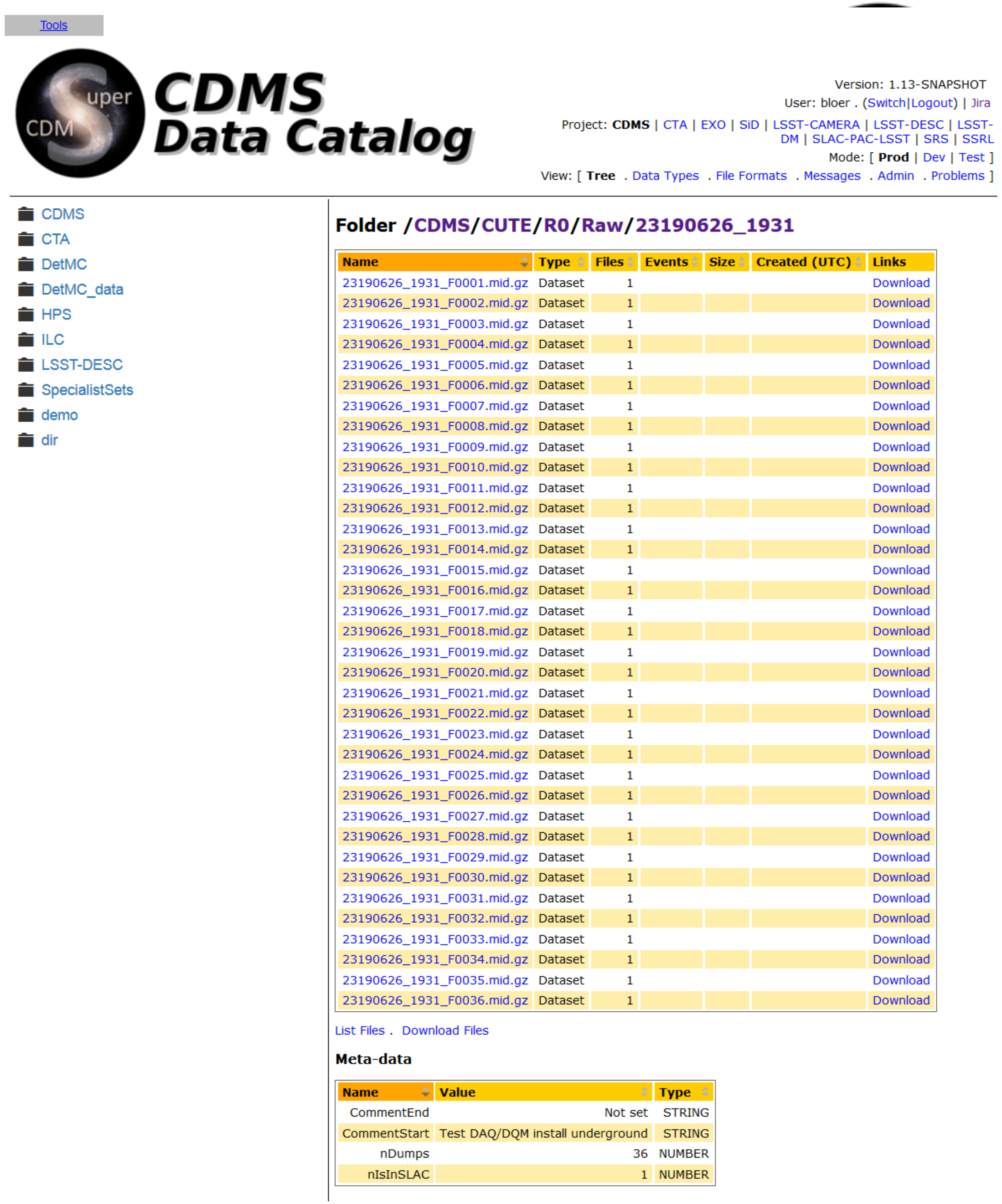
- GEANT4 particle transport
  - Calibration sources, radioactive backgrounds, fast neutrons, cosmic rays
- Custom code to simulate charge and phonon propagation in crystals
  - Initially written in Matlab
  - Newer GEANT4 Condensed Matter Package (G4CMP) to replace Matlab soon
  - Thousands of times more computation required than particle transport
- Custom code to simulate analog electronics and digitization
- Output converted to raw data format
  - Can include sampling noise from particular dataset
  - Use same reconstruction as raw data

Reconstruction

- C++, requires ROOT, BOOST, BLAS
  - Now transitioning to cmake due to BLAS being so annoying
- 3 stages, 2 output files per run:
  - Calculate noise PSDs for each channel during run (~3 hours)
  - Do slow calculations (e.g. N-dim Optimum Filter fits)
  - Do fast calculations (e.g. apply calibrations to convert from SQUID amps to keV)
- Each file contains a TTree for global quantities and one Ttree for each detector
  - Loading all data at once requires “friending” across 2(N+1)=50 trees
    - But most analysis is done detector-by-detector, and only uses stage 3 ntuples

Data Catalog

- Service developed by SLAC for LSST
- Browser and REST API, SQL under the hood
- Define arbitrary metadata for each data type
- Tracks locations and metadata only; does not manage data transfer, etc
- Scanner process at SLAC verifies file locations, etc



Tools

Super CDMS Data Catalog

Version: 1.13-SNAPSHOT
User: bloer . (Switch|Logout) | Jira
Project: CDMS | CTA | EXO | SiD | LSST-CAMERA | LSST-DESC | LSST-DM | SLAC-PAC-LSST | SRS | SSRL
Mode: [Prod | Dev | Test]
View: [Tree . Data Types . File Formats . Messages . Admin . Problems]

CDMS
CTA
DetMC
DetMC_data
HPS
ILC
LSST-DESC
SpecialistSets
demo
dir

Folder /CDMS/CUTE/R0/Raw/23190626_1931

Name	Type	Files	Events	Size	Created (UTC)	Links
23190626_1931_F0001.mid.gz	Dataset	1				Download
23190626_1931_F0002.mid.gz	Dataset	1				Download
23190626_1931_F0003.mid.gz	Dataset	1				Download
23190626_1931_F0004.mid.gz	Dataset	1				Download
23190626_1931_F0005.mid.gz	Dataset	1				Download
23190626_1931_F0006.mid.gz	Dataset	1				Download
23190626_1931_F0007.mid.gz	Dataset	1				Download
23190626_1931_F0008.mid.gz	Dataset	1				Download
23190626_1931_F0009.mid.gz	Dataset	1				Download
23190626_1931_F0010.mid.gz	Dataset	1				Download
23190626_1931_F0011.mid.gz	Dataset	1				Download
23190626_1931_F0012.mid.gz	Dataset	1				Download
23190626_1931_F0013.mid.gz	Dataset	1				Download
23190626_1931_F0014.mid.gz	Dataset	1				Download
23190626_1931_F0015.mid.gz	Dataset	1				Download
23190626_1931_F0016.mid.gz	Dataset	1				Download
23190626_1931_F0017.mid.gz	Dataset	1				Download
23190626_1931_F0018.mid.gz	Dataset	1				Download
23190626_1931_F0019.mid.gz	Dataset	1				Download
23190626_1931_F0020.mid.gz	Dataset	1				Download
23190626_1931_F0021.mid.gz	Dataset	1				Download
23190626_1931_F0022.mid.gz	Dataset	1				Download
23190626_1931_F0023.mid.gz	Dataset	1				Download
23190626_1931_F0024.mid.gz	Dataset	1				Download
23190626_1931_F0025.mid.gz	Dataset	1				Download
23190626_1931_F0026.mid.gz	Dataset	1				Download
23190626_1931_F0027.mid.gz	Dataset	1				Download
23190626_1931_F0028.mid.gz	Dataset	1				Download
23190626_1931_F0029.mid.gz	Dataset	1				Download
23190626_1931_F0030.mid.gz	Dataset	1				Download
23190626_1931_F0031.mid.gz	Dataset	1				Download
23190626_1931_F0032.mid.gz	Dataset	1				Download
23190626_1931_F0033.mid.gz	Dataset	1				Download
23190626_1931_F0034.mid.gz	Dataset	1				Download
23190626_1931_F0035.mid.gz	Dataset	1				Download
23190626_1931_F0036.mid.gz	Dataset	1				Download

List Files . Download Files

Meta-data

Name	Value	Type
CommentEnd	Not set	STRING
CommentStart	Test DAQ/DQM install underground	STRING
nDumps	36	NUMBER
nIsInSLAC	1	NUMBER

End-user analysis

- Planning structure now
- Previous model: most users logged into a single server at Stanford, ran interactive Matlab sessions
- If possible, want to keep to mostly-interactive analysis (i.e. not have to submit jobs for absolutely everything)
- Current plan to push everyone to python, load our ROOT data into pandas DataFrames
- Hope to use JupyterHub to provide environment, but experiencing problems with our current implementation

Software Challenges

Challenge: Local data store

- Ntuples available at Site A
- Users want to do (interactive) analysis at Site B
- On demand, copy (many) files from Site A to Site B
  - Sync/lock mechanism in case multiple users try to copy same files at the same time
- Persistent file store, not copy one file at a time to scratch
  - Allocate a quota, and refuse or do cleanup if transfer too large
  - (Automatically) remove oldest accessed files to free up space

Challenge: IO reduction (skims)

- During early analysis, workflow is:
  load whole ntuples -> develop basic cuts (e.g. threshold, reconstruction χ^2)
- Once basic cuts exist, want to apply cuts before loading data
  - Skip reading events just to immediately discard
  - Skip having to read otherwise-unneeded branches (e.g. chisq)
- Skims (copy of files with only events passing cuts) are efficient, but
  - Cuts tend to be tweaked over time, leading to lots of copies
  - What combinations of cuts to create skims for? New cuts added constantly
- Something like ROOT TEntryList that only reads listed entries would be ideal

Challenge: Horizontal chunk synchronization

- For much of analysis, each detector is treated independently, e.g. quality cuts
- Each detector’s ntuples stored in a separate tree
- After thresholds, skims, etc, trees contain different sets of events
- How to match events to do cross-detector searches, e.g. for crosstalk or coincident alpha/daughter hits?
  - Database: joins
  - ROOT: TTreeIndex
  - Pandas: join on DataFrame index if dataset fits into memory

DetA	DetB
1	2
4	3
5	4
9	7
12	11
13	14
14	16
16	18
18	22
24	24
28	27
29	28
32	29

DetA	DetB
1	-
-	2
-	3
4	4
5	-
-	7
9	-
-	11
12	-
13	-
14	14
16	16
18	18
-	22
24	24
-	27
28	28
29	29
32	-

DetA	DetB
4	4
14	14
16	16
18	18
24	24
28	28
29	29

Challenge: Horizontal chunk synchronization

- For large datasets, uproot and root_pandas both iterate over smaller chunks of data
- BUT: if chunk boundaries misaligned, not all events present for join

DetA	DetB
1	2
4	3
5	4
9	7
12	11
13	14
14	16
16	18
18	22
24	24
28	27
29	28
32	29

DetA	DetB
1	2
4	3
5	4
9	7

DetA	DetB
12	11
13	14
14	16
16	18

DetA	DetB
18	22
24	24
28	27
29	28

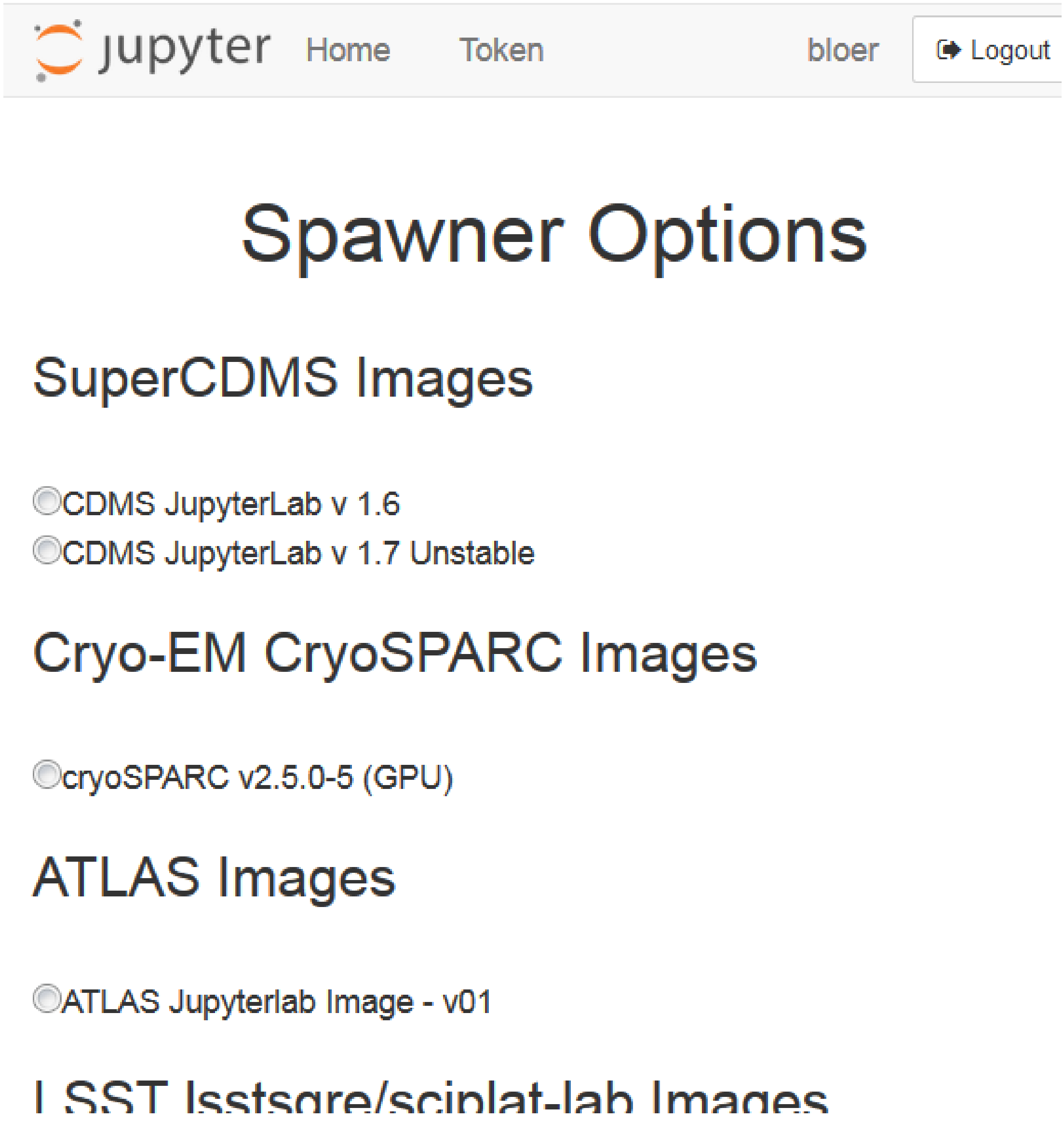
DetA	DetB
32	29

Challenge: Global software releases

- Multiple software packages with complex dependencies
  - E.g. recon v2.1 handles new feature of simulation v5.3 that requires geant v10.6
- Want single CDMS global software version to fully specify everything for documentation and reproducibility
- Produce and distribute binaries for multiple platforms (SL6, CentOS7) via cvmfs
- Old and new releases side-by-side, without multiple copies of packages with slower release cycle
  - But also need to keep track of breaking dependencies, e.g. ROOT
- Currently being built on a rickety tower of symlinks and custom bash scripts…

Challenge: resource management on interactive nodes

- SLAC hosts an experimental JupyterHub server
  - Web interface to spawn per-user JupyterLab instance
  - Environment defined by Docker image: allows backward compatibility
- Heavy use; frequent instability from lack of memory, etc
- Exploring self-hosting, dedicated hardware, etc



Challenge: Ease of access to member institution computing resources

- Many CDMS member institutions have local small-to-medium sized computing clusters
- Occasionally want to leverage all of it at once
  - e.g., reprocess 3 years’ worth of raw data with new/improved algorithm
- Handled by manually dividing jobs into separate lists, each managed by a user on their local cluster
- Explored DIRAC to have a central interface to all computing resources, but lack the manpower to implement it

DIRAC THE INTERWARE

Thank you