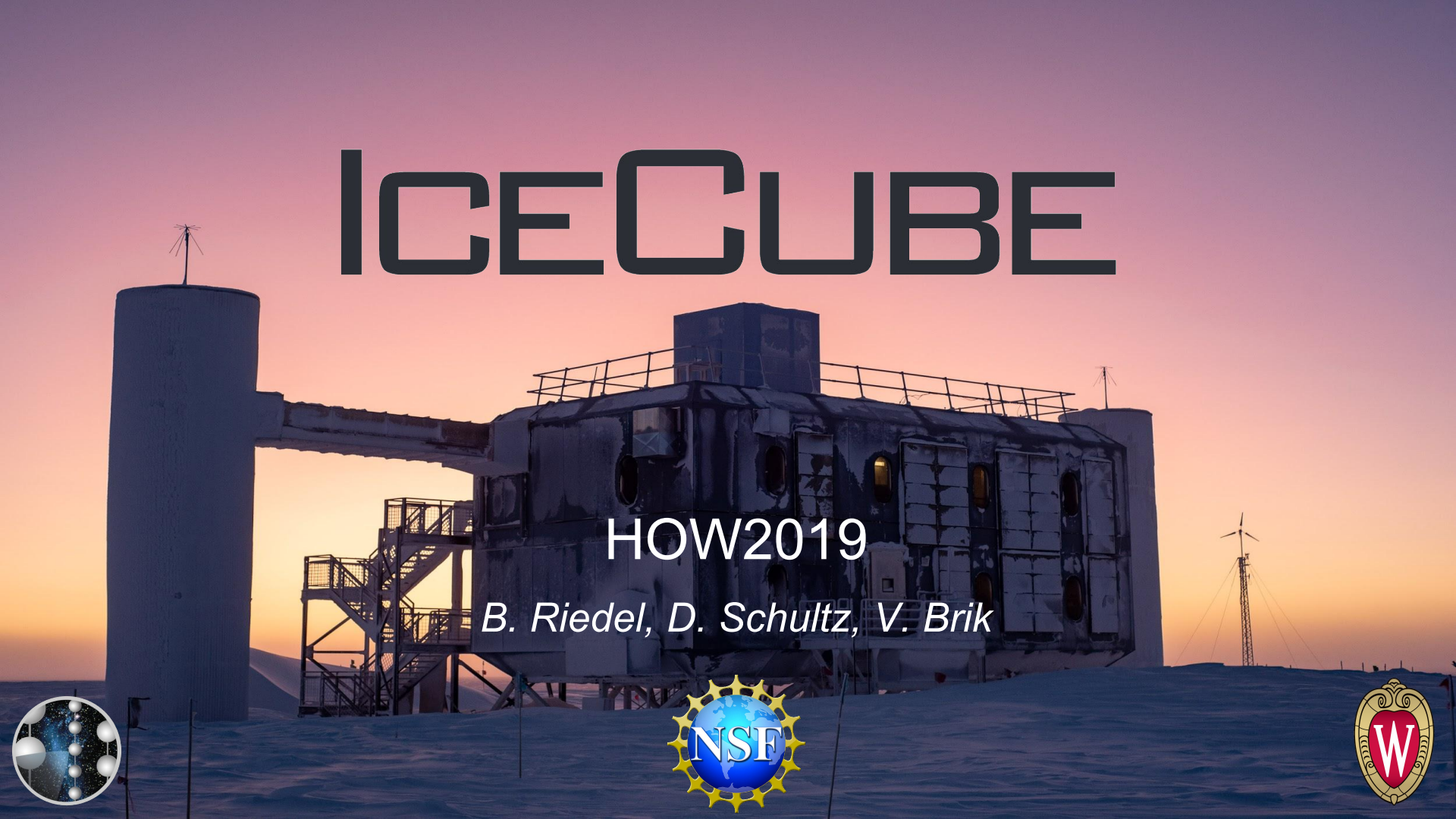
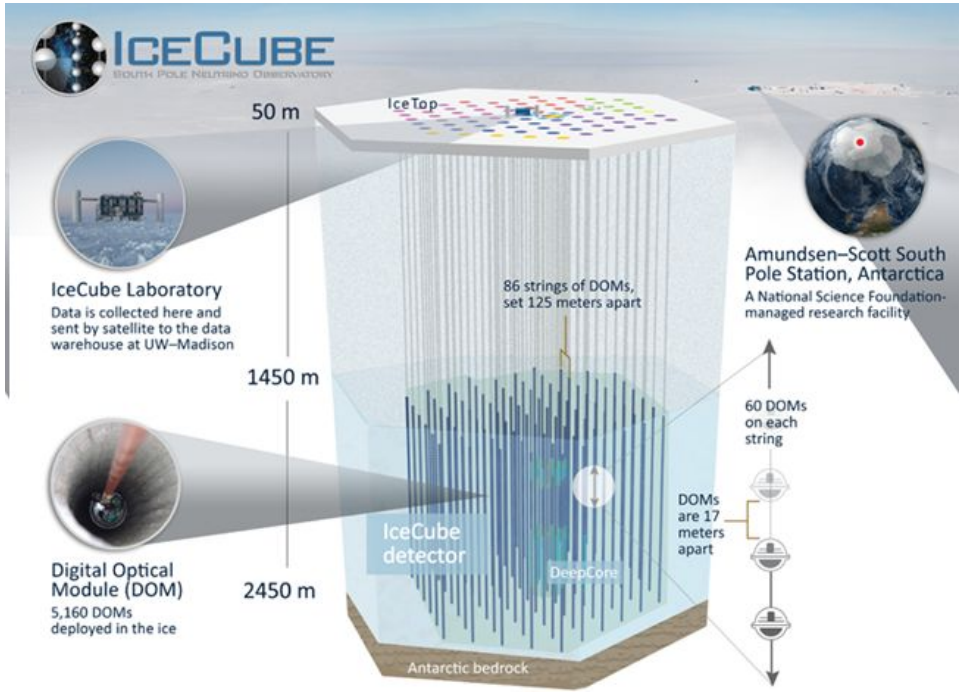
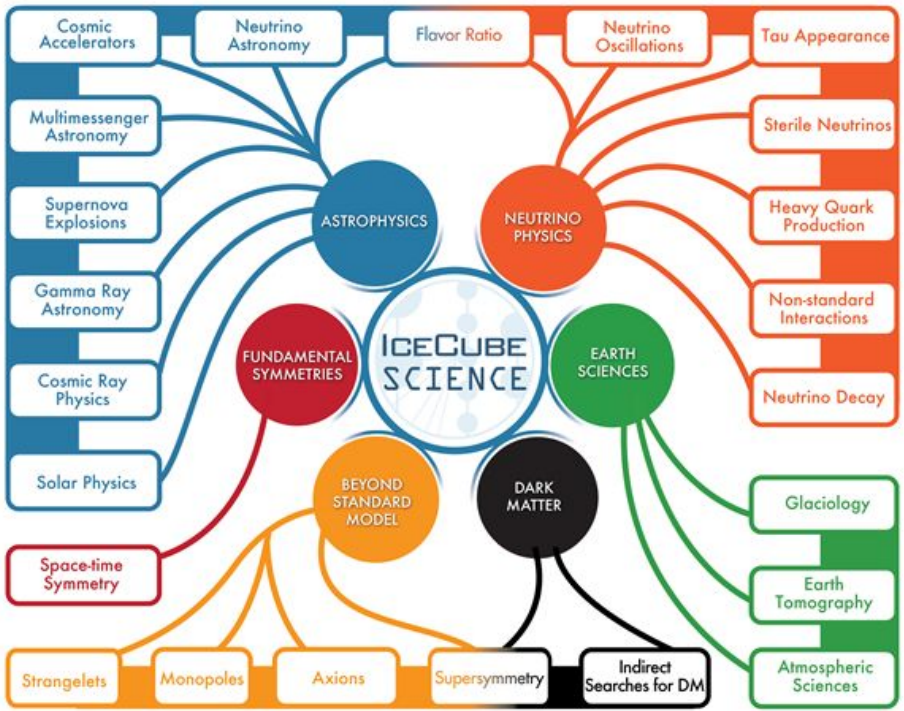

IceCube
HOW2019
B. Riedel, D. Schultz, V. Brik
NSF

Outline

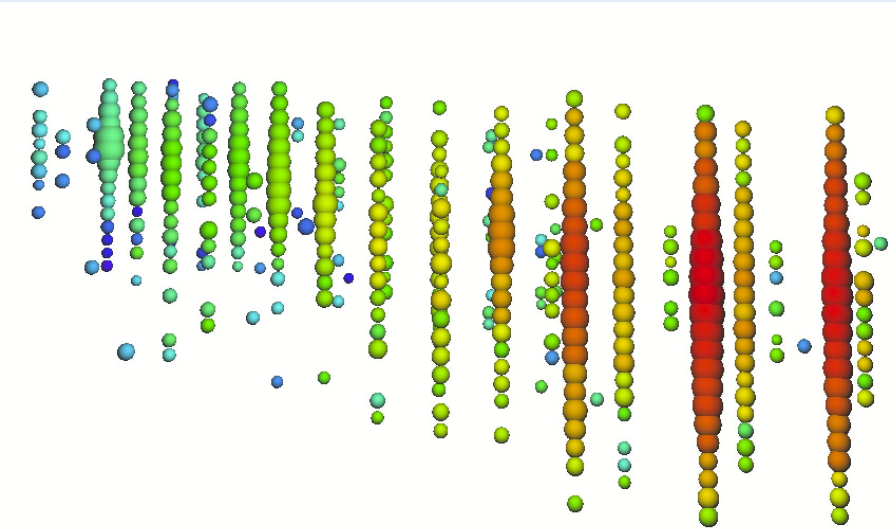
- IceCube Computing
- Running Jobs - Grid status
- Simulation Developments
- Data Management - file metadata catalog, long term archive
- Looking Forward: CESAR, E-CAS, CSSI, ...

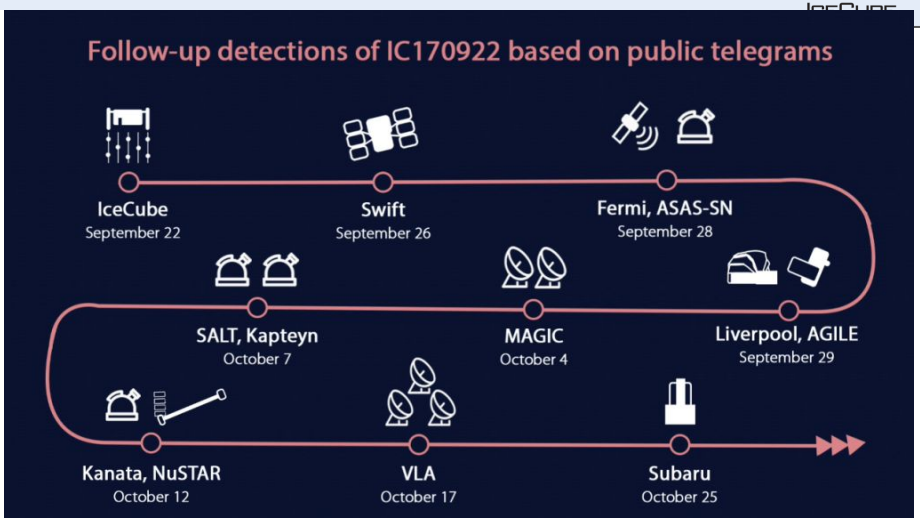
IceCube Detector and Science





Multi-Messenger Astrophysics





IceCube detected an event that came from Blazar TXS 0506+056

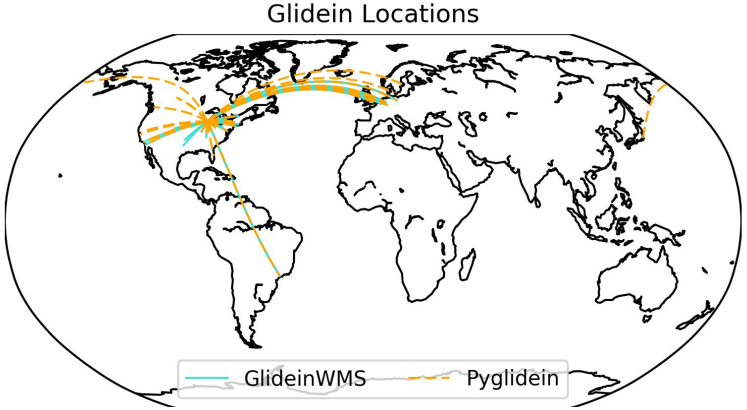
Follow up observations by several telescopes showed signal

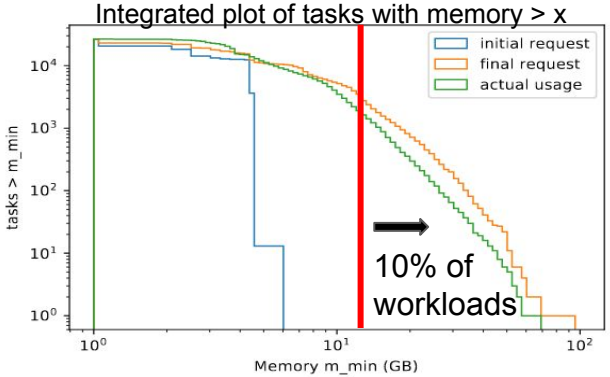
Back catalog showed access for this source

NS+NS merger would be ideal IceCube+LIGO observation

IceCube Computing

- Global heterogeneous resources pool
- Mostly shared and opportunistic resources
- Atypical resources requirements and software stack
  - Accelerators (GPUs)
  - Broad physics reach - Lots of physics to simulate
  - Data flow includes leg across satellite
  - “Analysis” software is produced in-house
    - “Standard” packages, e.g. GEANT4, don’t support everything or don’t exist
    - Niche dependencies, e.g. CORSIKA (air showers)
  - Detector up time at 99+% level, build in natural medium
- Significant changes of requirements over the course of experiment - Accelerators, Multimessenger Astrophysics, alerting, etc.

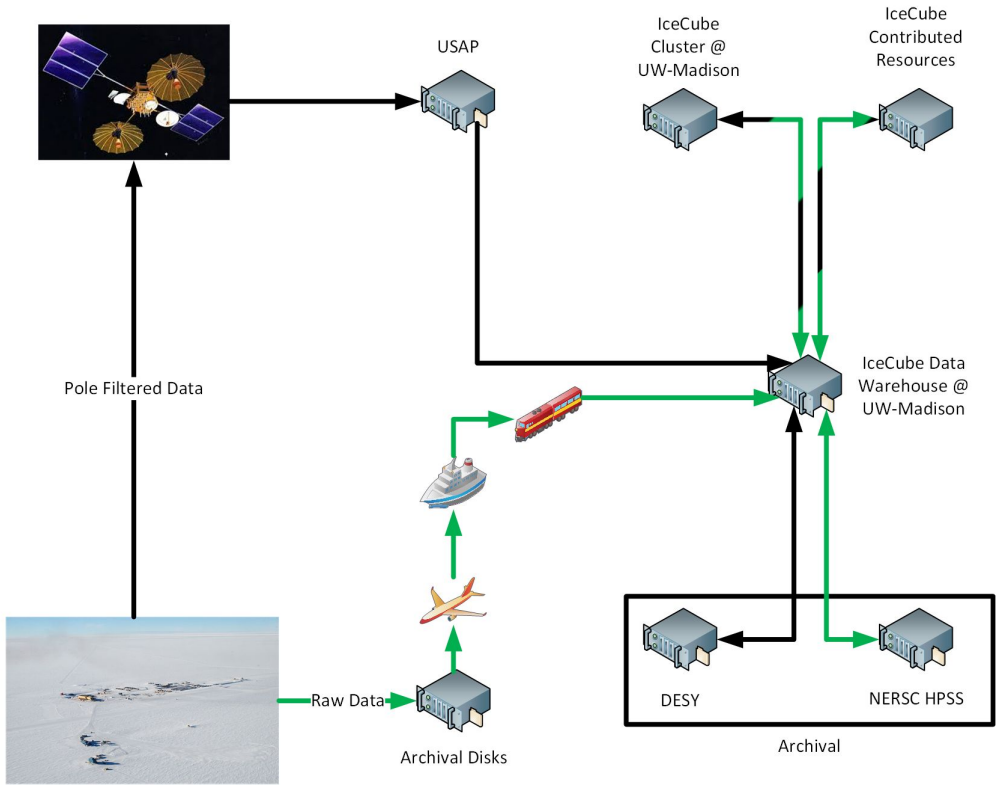




* All IceProd 2 tasks as of April 2018

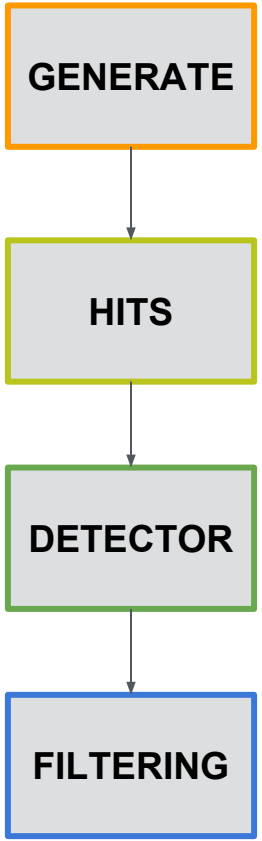
Data Flow and Processing

- Pole Filtered Data arrives via satellite - Arrives at UW-Madison and is reduced further to higher levels
- Raw data is written to archival disk at pole, retrieved once a year
- Raw data is archived at National Energy Research Scientific Computing Center (NERSC)
- Filtered data is archived at Deutsches Elektronen-Synchrotron (DESY)



Simulation Chain

Generate background (Cosmic Ray showers) and signal (neutrino interactions)
Propagate to detector



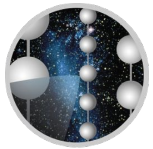
Photon propagation in ice - Requires GPUs

DOM hardware simulation
DAQ trigger emulation

Pole (Level 1) and offline (Level 2) reconstruction and filtering

Running Jobs

Grid Status



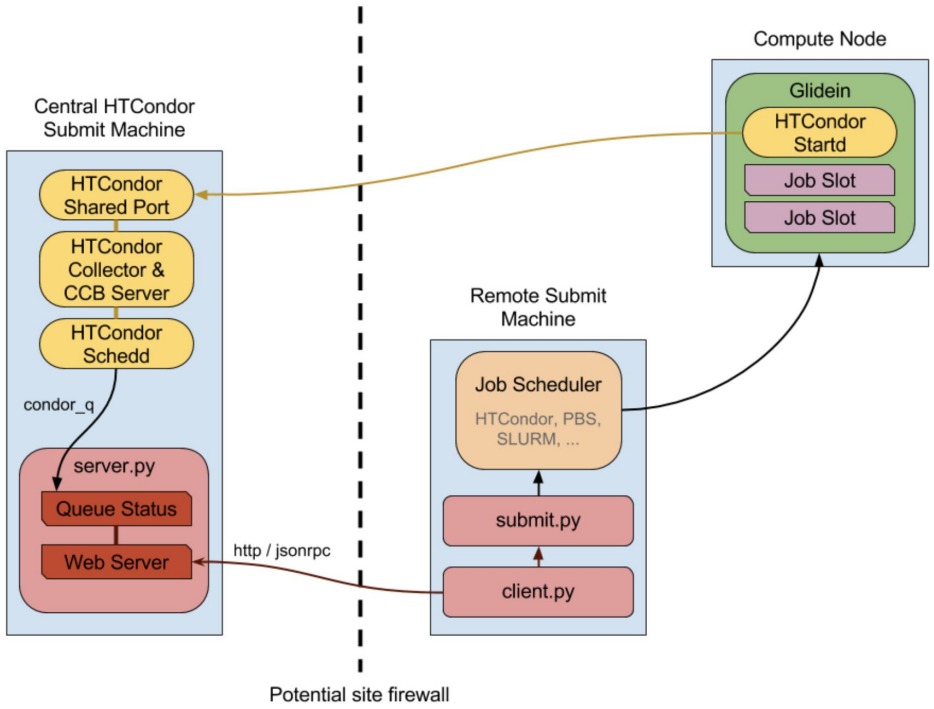
Pyglidein

A python server-client pair for submitting HTCondor glidein jobs on remote batch systems.

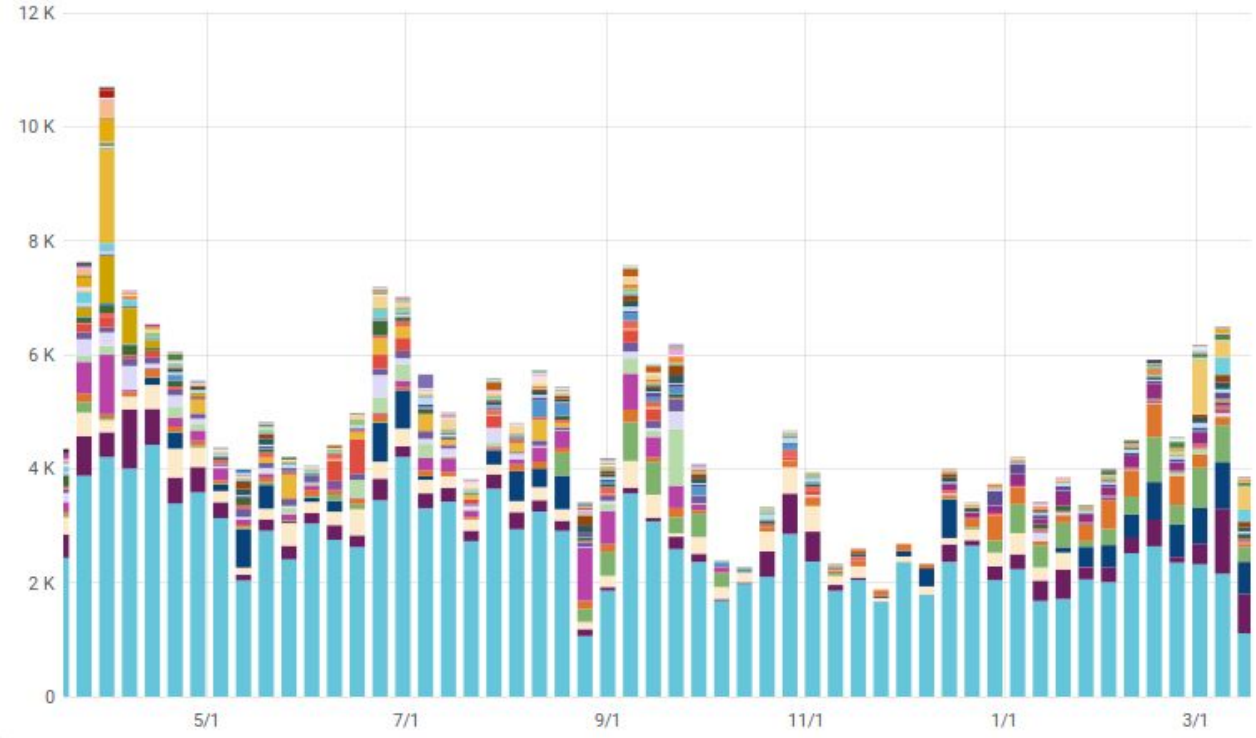
https://github.com/WIPACrepo/pyglidein

Motivation / requirements:

- MFA
- Lightweight library for easy remote operation
- UNIX philosophy

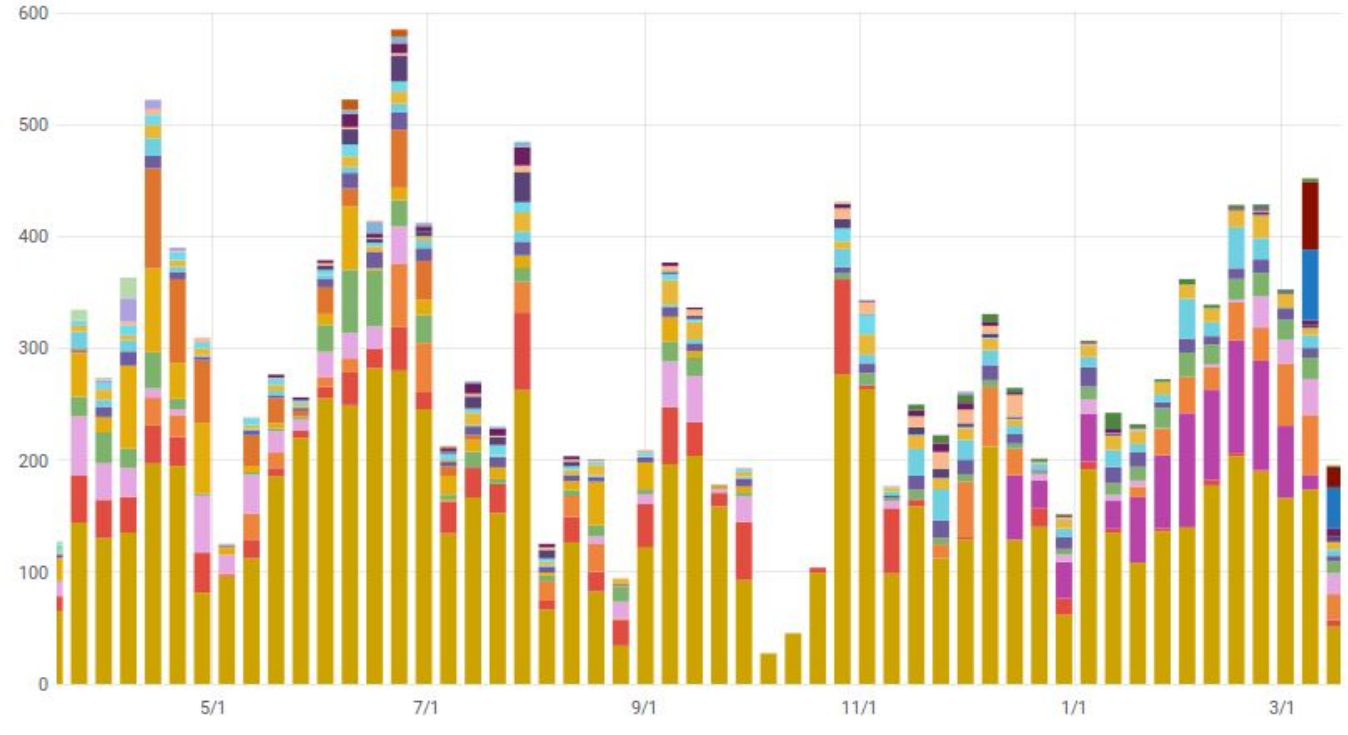


CPU usage per week



	max	avg	current
NPX	4.419 K	2.636 K	1.112 K
DESY-ZN	1.143 K	295	687
T2B_BE_IIHE	506	217	0
DESY	819	200	567
UMD	783	170	261
LIDO_Dortmund	589	141	61
msu	1.044 K	136	0
GPGrid	996	71	0
UKI-NORTHGRID-MAN-HEP	420	70	19
CHTC	208	58	22
Bridges	596	57	46
USCMS-FNAL-WC1	411	47	0
NWICG_NDCMS	136	42	3
RWTH-Aachen	250	40	65
MSU	255	38	0
Nebraska	282	35	16
Cedar	830	35	0

GPU usage per week



	max	avg	current
NPX	282	153	51
GZK	85	19	6
MSU	102	14	0
DESY	56	14	23
UMD	52	12	19
Bridges	56	12	11
msu	75	11	0
xstream	90	8	0
Marquette	17	8	5
Crane	37	7	6
Comet	21	7	7
T2B_BE_IIHE	18	3	0
SU-ITS-CE3	27	3	4
UCSDT2	20	3	0
SU-ITS-CE2	16	3	8
Illume	63	2	36
SDSC-PRP	61	1	18

XSEDE 2018 allocation

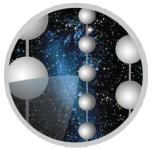
June 2018 - start of our 3rd research allocation. Target GPUs

Represents ~15% of our GPU capacity at UW-Madison

A good opportunity to exercise integration of supercomputers and extended period operations

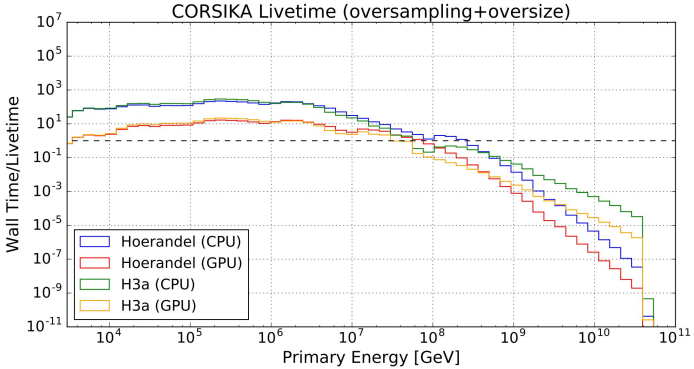
System	***Service Units awarded***	***GPU nodes***	***Used so far***
Bridges GPU	*287,251*	*16x2 K80 + 32x2 P100*	*~58%*
Comet GPU	*180,000*	*36x4 K80 + 36x4 P100*	*~56%*
OSG	*4,000,000*		*>100%*

Simulation Developments



Physics Software — CORSIKA

CORSIKA Livetime Issues



- *A priori* simulation doesn't know if a shower is "interesting" to IceCube - It can take over **500x** the compute time to get the desired livetime
- Even more problematic for generating air showers with $<10^3$ GeV primaries - Scientifically interesting, yet production wasteful - Products don't trigger detector, so resources are "wasted"
- Single muon simulation (MuonGun) is much faster, but introduces systematics (muon bundles)
- Analyzers would prefer CORSIKA, not possible by brute force simulation

Physics Software – CORSIKA

CORSIKA Dynamic Stack

- D. Baack (Dortmund), J.van Santen (DESY), K. Meagher (WIPAC)
- Better control shower generation
  - Kill showers as early as possible
  - Save CPU and GPU time
- Initial simple settings show factor of 2 reduction in CPU across all energy ranges.

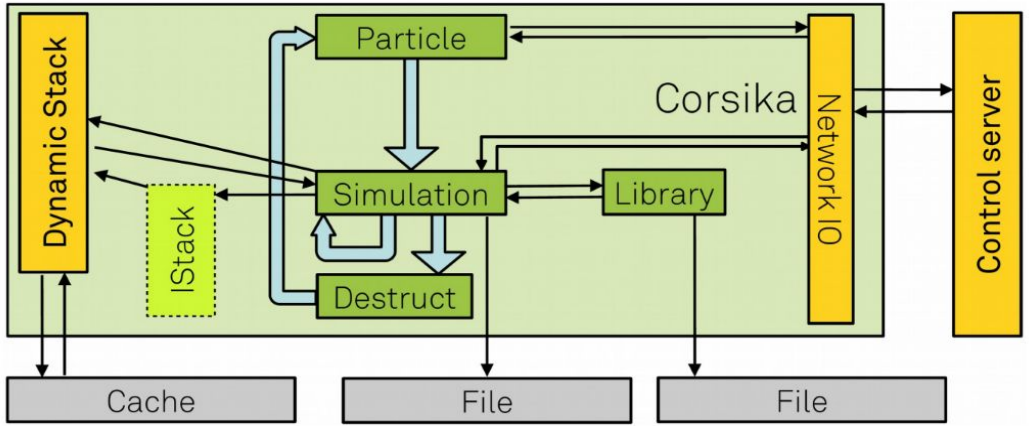
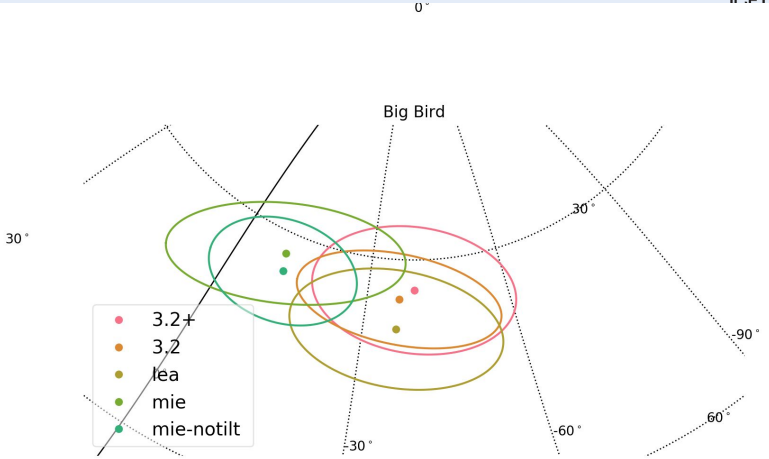


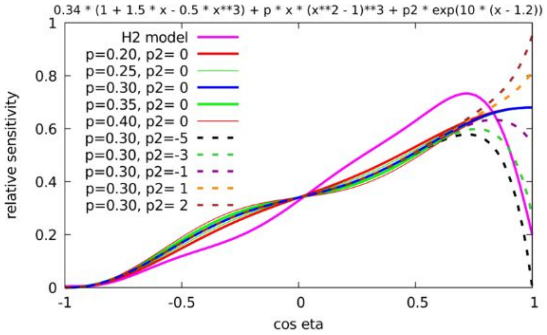
*Image from D. Baack (Dortmund)

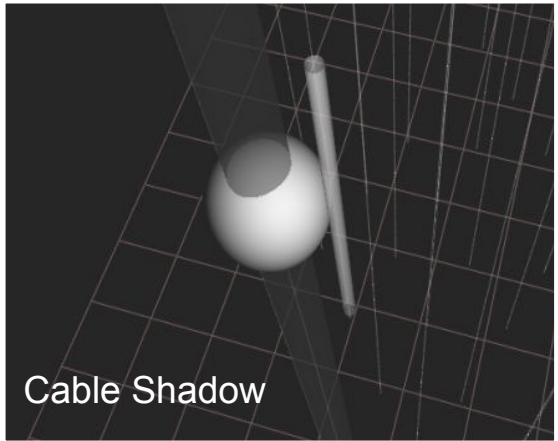
Physics Software — Photon Propagation

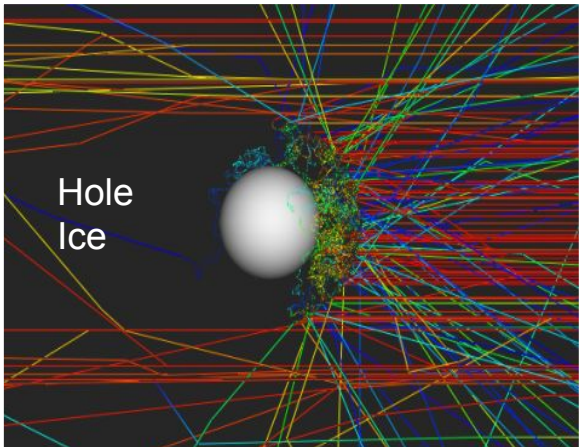
Ice model uncertainties

- Modeling *in situ* the proper angular and overall acceptance of DOMs is an extremely hard problem - Ice is a natural medium
- Important systematic effect

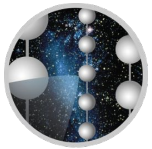








Data Management



File (metadata) Catalog

Motivation: various services that “handle files”

- IceProd, JADE, Long Term Archive (LTA), …
- Each doing independent bookkeeping in internal DBs

Goal: Collect all file metadata in a central catalog

- Build a central catalog that contains all file metadata - Users could query it to find data files
- Can be used by all these services - Keep one consistent view of the metadata for all IceCube files

File (metadata) catalog

Structure

- Python web server with REST API - Speaking JSON
- MongoDB backend

Schema

- Core metadata (uuid, name, location, checksum, …)
- Specific metadata in sub-objects; can have more than one

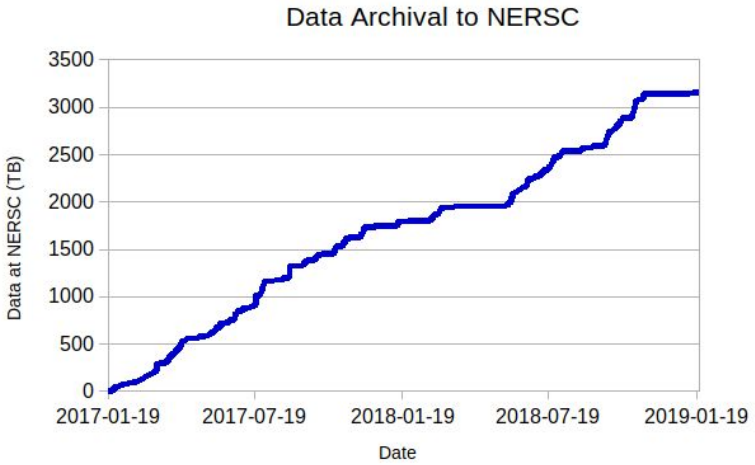
Exposes a few useful queries

- Files in a simulation dataset, good files in an IceCube run, files that contain a specific Event ID,...

JADE Long Term Archive

JADE Long Term Archive software components:

- Indexer - Metadata collection and validation
- Bundler - Creates large (>500GB) archives
- Mirror - Transfer via Globus from UW-Madison to DESY/NERSC



JADE Long Term Archive - Issues

Operations are still very labor-intensive

- Indexing/Bundling - Submitting HTCondor jobs to the local cluster
- Transfer - Transfers scheduled as bundle files are produced
- Taping - Moving archival bundle at NERSC from disk to HPSS

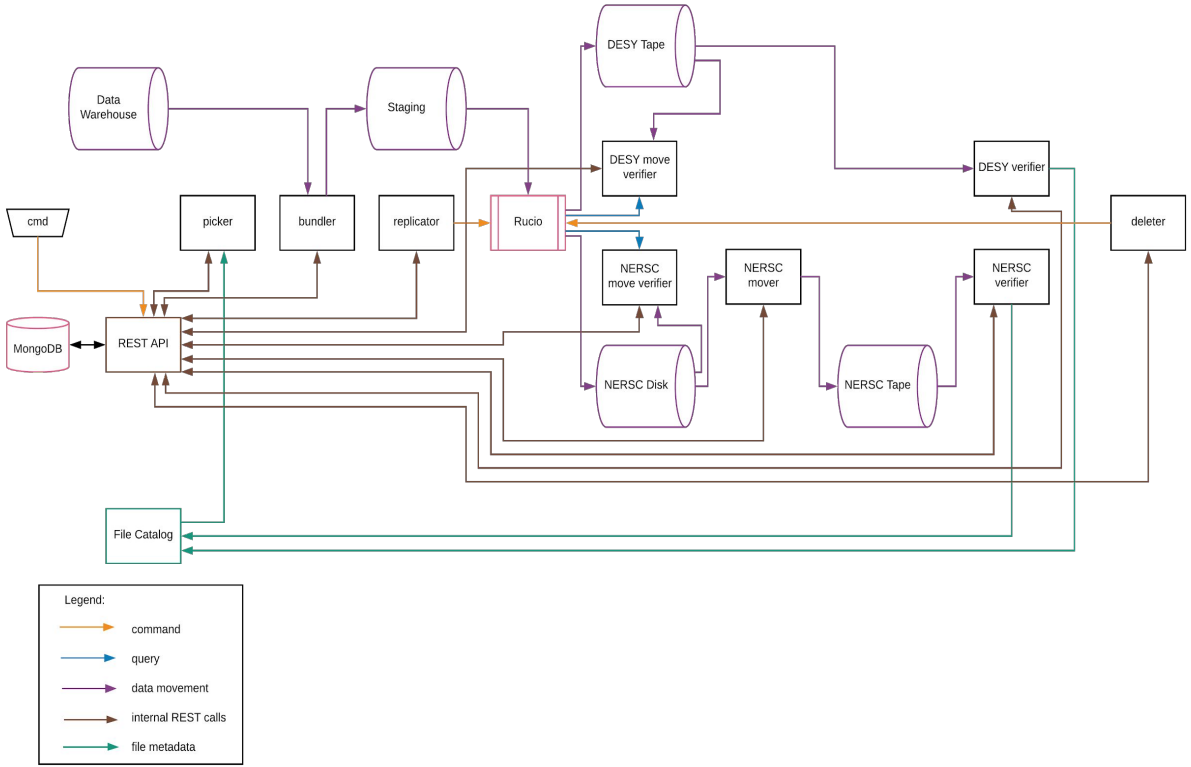
Reporting Tools - Primitive and buggy

Globus.org going closed-source, paid - Question about what to do at NERSC, DESY uses dCache

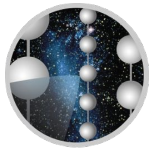
New Long Term Archive

Redesigned to integrate better with other services

- Use already proven code - Rucio for data transfer orchestration
- Integrates with IceCube File Catalog
- Better monitoring / metrics



Looking Forward



Future Plans

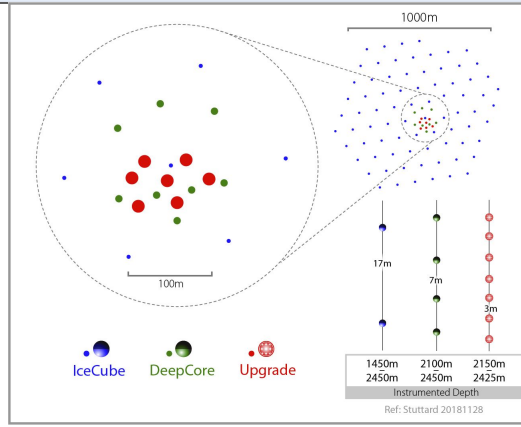
Detector upgrades starting 2021/22 and could go until 2030s

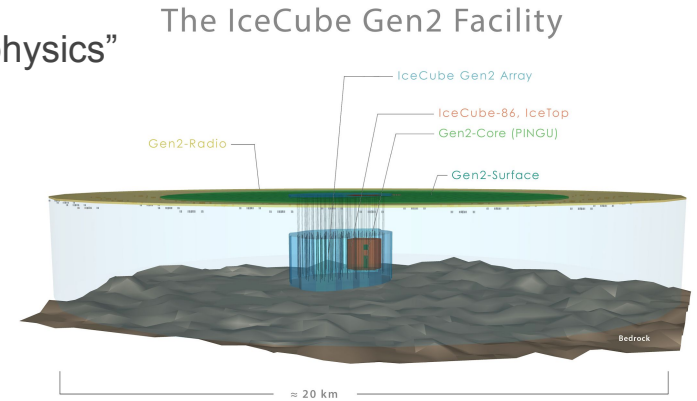
Bottlenecks on several fronts:

- GPU resources
  - More coming online, but possibly hard to access (TITAN, Summit)
  - Machine learning will require more
- Ever changing physics and detector knowledge
  - Ice - Birefringence? Hole ice properties?
  - Cosmic Ray interaction model
- “Small” community driven by individual experiments
- Limited personpower to drive development - Further away from “physics” the less developers

What to do?

- Scalable Cyberinfrastructure to support Multi-Messenger Astrophysics (SCIMMA)?
- WLCG for MMA?





E-CAS

New cloud initiative with Internet2 funded by NSF (Phase 1 and 2) and cloud providers (Phase 1 - AWS, GCP, Oracle)

- Realtime event processing bursting into AWS - Multi-messenger astrophysics with LIGO, EM followup
- Ice model refinements using one-time GPU resources on GCP
- JupyterHub for deep learning on GCP GPUs and TPUs, and AWS FPGAs

GPUs - Midscale Research Infrastructure

Submitted Midscale Research Infrastructure 1 pre-proposal for a GPU cluster at SDSC with UCSD cryoEM group

- 720 NVIDIA T4, 288 NVIDIA V100 - Even split between science groups
- 10 PB Lustre for cryoEM group
- Integrated into national cryoEM facilities (CA, NY, OR, CA) and cryoEM archival (NERSC)
- Piece to meet goal of IceCube advisory panel - “Increase computing by a factor 10 in 18 months”

Possible Proposals

Considering CSSI and/or I-DIRSE-FW proposal to address several open issues in IceCube workflows, data management, and software development

- Improved realtime results from IceCube
- Fast follow-up on alerts from other observatories in current and archival data
- Improvements to IceCube workflows - Intelligent scheduling
- “Sanity Checks” on IceCube software
- New data management in IceCube - “No more POSIX access”

Summary

Computing: GPUs now nearing steady state in high volume
Supercomputers: additional GPU capacity

Simulation Developments

Data management: new Long Term Archive + Rucio in progress

Proposals: multiple in the works

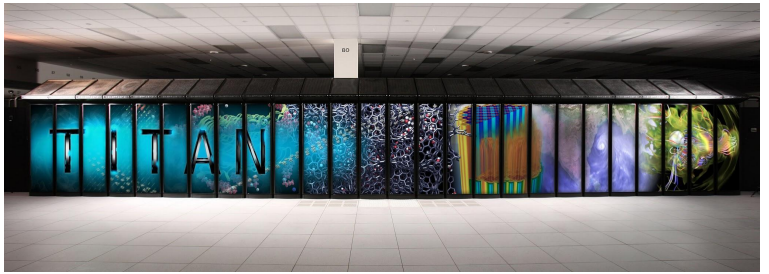
Thank You

Backup Slides

TITAN

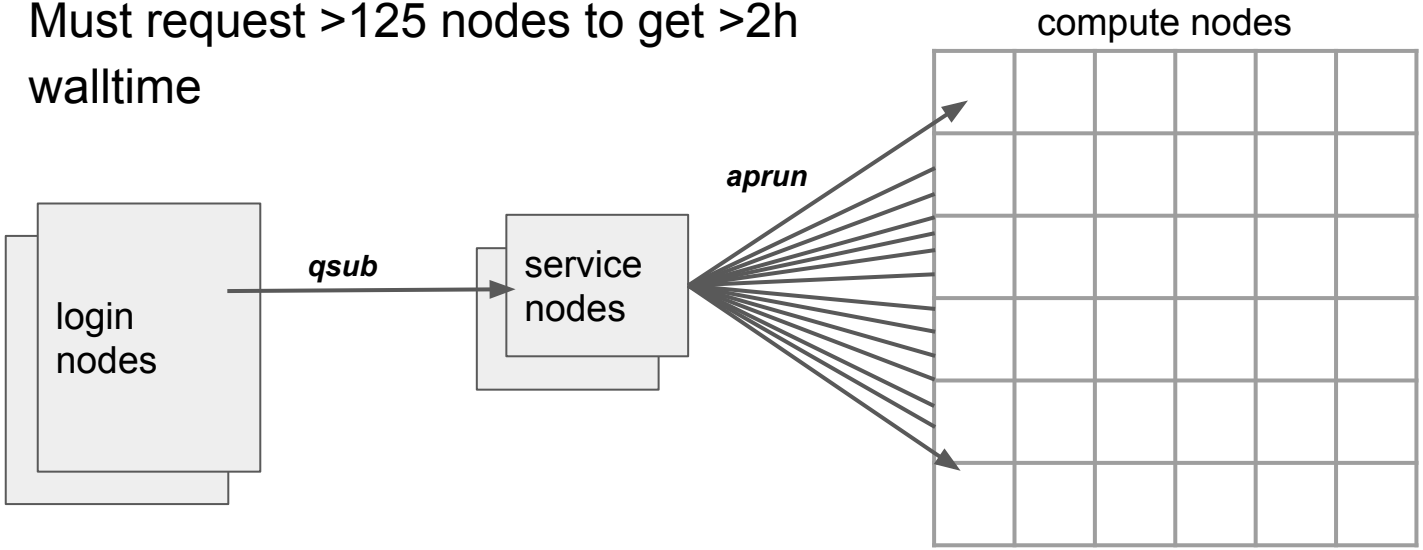
Cray XK7 at Oak Ridge Leadership Computing Facility

- 18,688 physical compute nodes
  - 16-core AMD Opteron, 32GB RAM
  - 6GB nVidia Kepler GPU
- ~300,000 cores, 600 TB RAM, 18.7k GPUs
- “Cray Linux Environment”: regular linux on service nodes, Linux microkernel on compute



TITAN

- Execute nodes on isolated network
- Very MPI-oriented
- Policies adverse to smaller jobs
  - Only 2 concurrent jobs <=125 nodes
  - Must request >125 nodes to get >2h walltime



HTCondor on Titan - Singularity jobs

Singularity container with HTCondor and subset of IceCube software

- ~40 minutes to rebuild container from scratch and upload to Titan
  - annoying for small changes
  - load as much as possible from outside container during development
- sshd for interactive debugging since Titan doesn't provide it
- Few bash scripts scripts to start and manage condor pool

HTCondor on Titan: 1 Titan job = 1 Condor pool

Each node does real-time logging of resource usage (GPU, CPU, mem …)

Pool shuts down automatically a few minutes after running out of idle jobs (control cost, reduce wasted nodes)

- Pool resumed in a new job that may request different resources
- Condor state stored on shared file system

Pool can be monitored by sshing into central manager

Log files on shared filesystem

- Can be watched or analyzed from a login node

IceCube on Titan - Notes

Singularity support on Titan extremely, extremely useful

Development iterations on Titan can be slow. Debugging is inconvenient.

- Interactive access to running containers makes life less painful

HTCondor on TITAN works well

- So far, 45k simulations done (26% of allocation), 2.1 TB output

Working in collaboration with ATLAS experts to test using Panda for running IceCube jobs in TITAN

- Work in progress

Pyglidein - What is next

Use a newer version of HTCondor and Parrot to be able to use Singularity

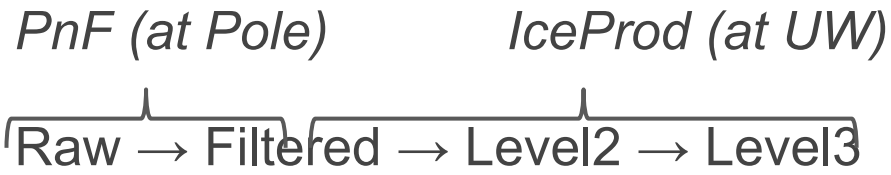
- same functionality as already present in GlideinWMS, so users can use Singularity without caring about the glidein “flavor”

Want to test new GPUsUsage feature in HTCondor 8.7.7

- need a patch in order to use it inside a glidein (has been already submitted to HTCondor)

Reproducibility/traceability

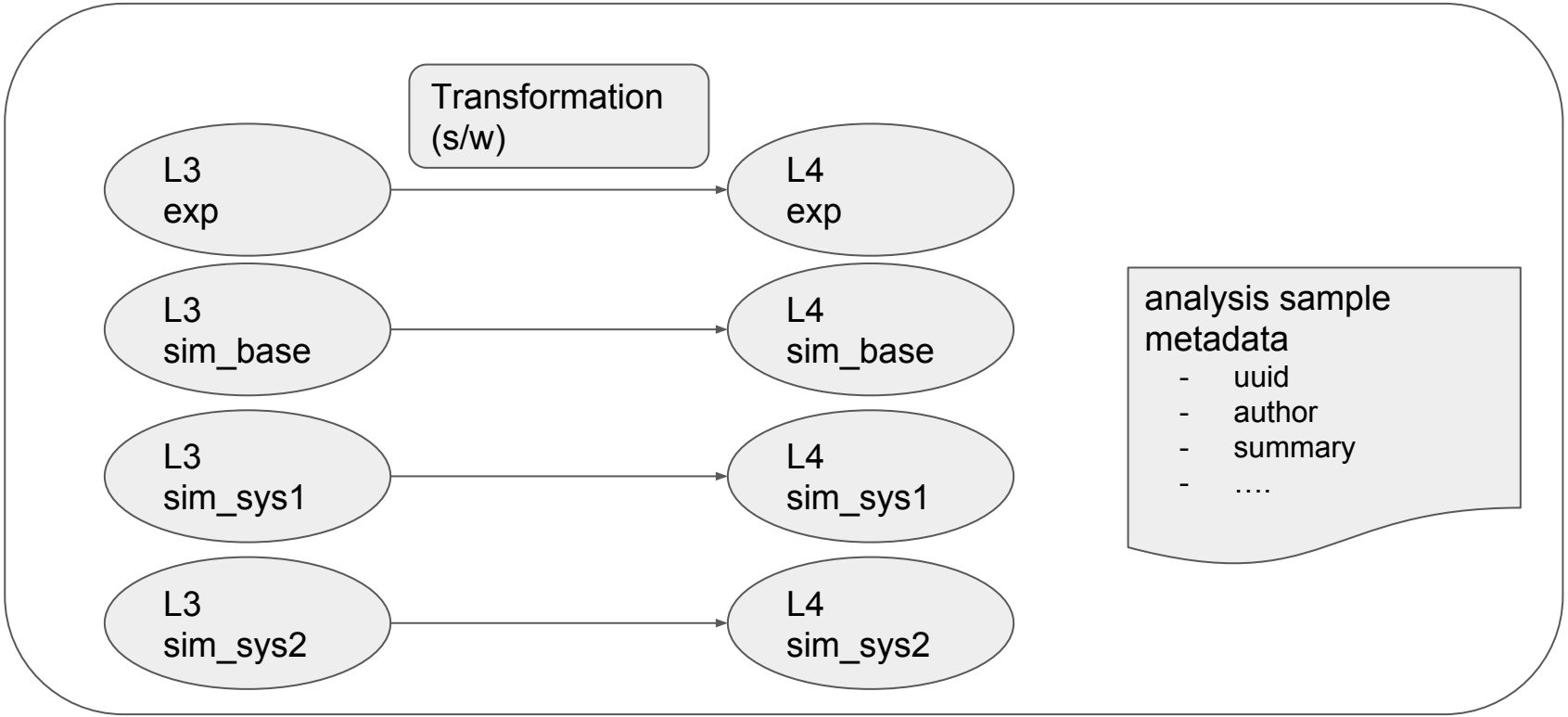
The IceCube data processing pipeline is centrally managed down to Level3



User analysis starts at Level3 or Level2. Generates various data products which are finally used to produce results in publications.

- Tracing the data provenance is a challenge
- Goal: being able to track which data files went into generating a particular scientific result (publication)

Reproducibility/traceability



Data Processing Framework

IceProd

IceProd - Dataset bookkeeping

What is it:

- Dataset submission framework
- Data provenance - Keeps track of metadata, software config, versioning
- Monitors job status, resource usage - Could be done as part of glidein infrastructure (more later ...)
- Retries failed jobs
  - Can resubmit with different requirements
  - Blacklist site automatically

IceProd - Dataset bookkeeping

What do we use it for:

- Simulation production
- Experimental data processing - Common (Level 2) and physics workgroup specific (Level 3) analysis
- Increasingly higher levels of common processing

Future Plans:

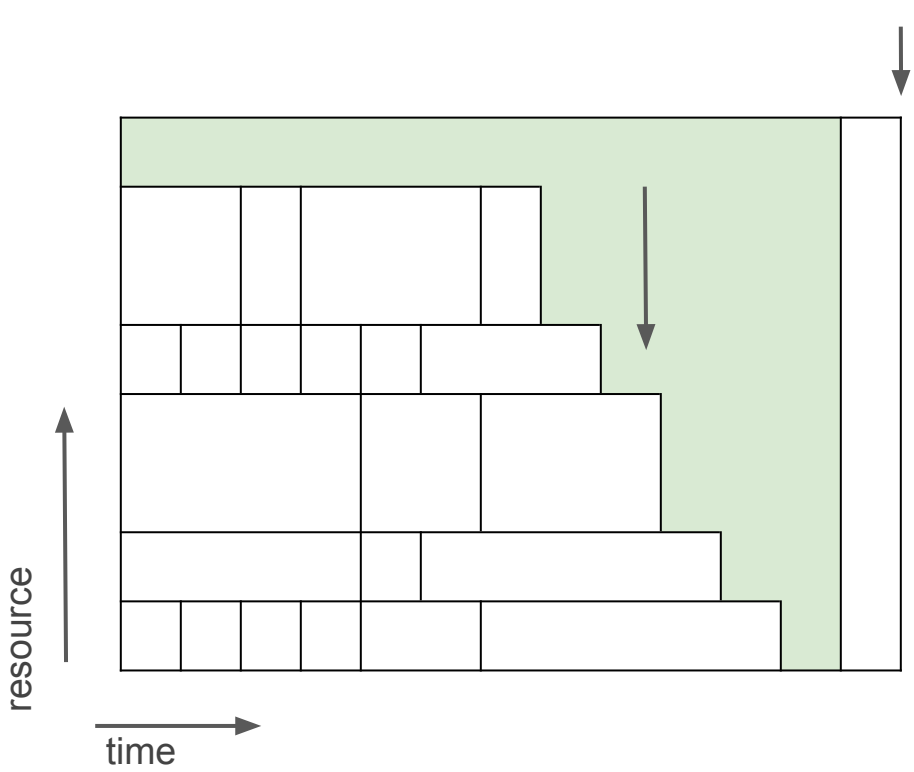
- User analysis

IceProd - Internal Pilot

We run a **pilot** inside the HTCondor job - Things it does / plans to do:

- Aggregate communications with the IceProd server
  - IceProd pilots are whole-node jobs: one communication link per node
- Resource monitoring in real-time
  - cpu, gpu, memory, disk usage, time
- Asynchronous file transfer
  - stage in/out files for next/prev jobs while jobs execute
- Dynamically resizable “jobs”

Dynamically resizable slots



give more resources to a long-running job to try and complete it before the end of the glidein