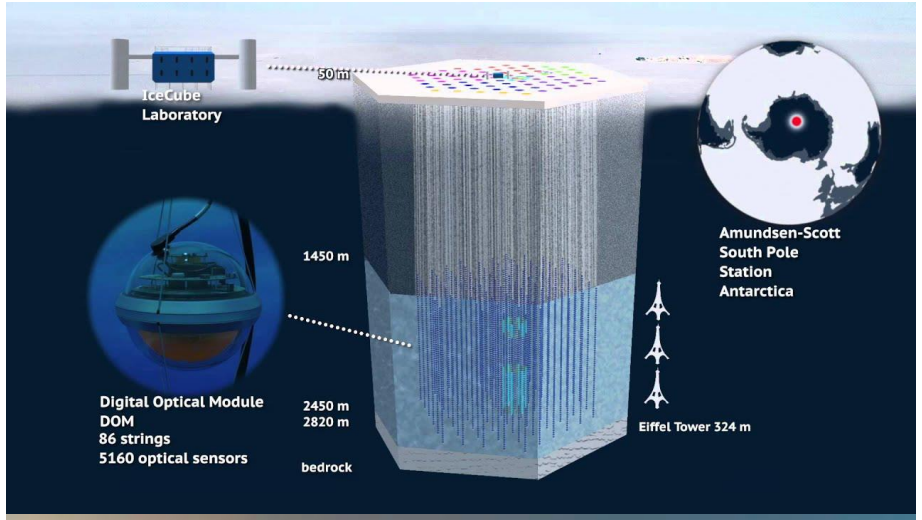
IceCube Glideins and Autonomous Pilots

Benedikt Riedel
UW-Madison

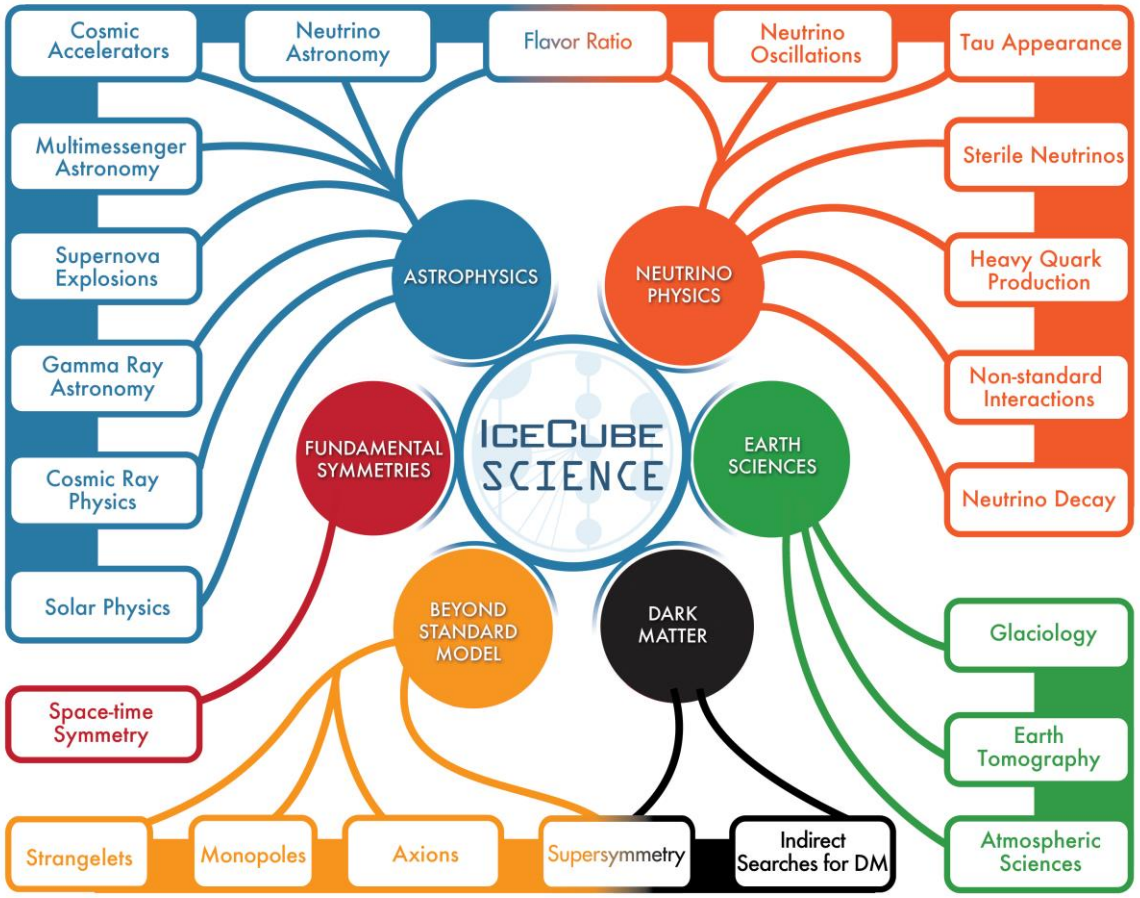
HTCondor Week 2021
May 27, 2021

IceCube



- A cubic kilometer of ice at the South Pole is instrumented with 5160 optical sensors
- South Polar Ice cap contains clearest known material on earth – Single photon can travel up to 300 m
- Detector is larger than the physical volume - Scales with energy of neutrino and neutrino type

Why are we doing this?



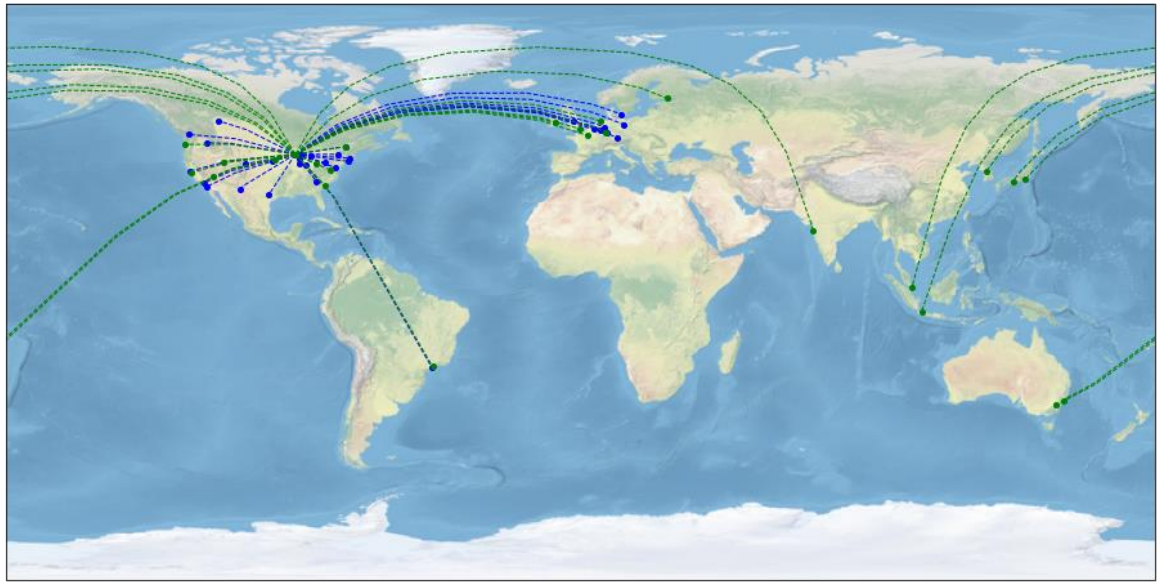
- Novel instrument in multiple fields
- Broad science abilities, e.g. astrophysics, particle physics, and earth sciences
- Lots of data that needs to be processed in different ways
- Lots of simulation that needs to be generated

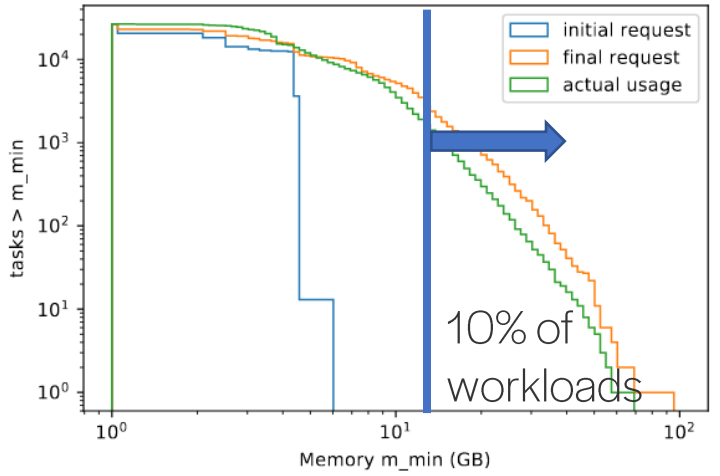
IceCube Computing – 30000 Foot View

- Classical Particle Physics Computing
  - Ingeniously parallelizable – Grid Computing!
  - "Events" - Time period of interest
  - Number of channels varies between events
  - Ideally would compute on a per event-basis
- Several caveats
  - No direct and continuous network link to experiment
  - Extreme conditions at experiment (-40 C is warm, desert)
  - Simulations require "specialized" hardware (GPUs)
  - In-house developed and specialized software required
  - Large energy range cause scheduling difficulties – Predict resource needs, run time, etc.

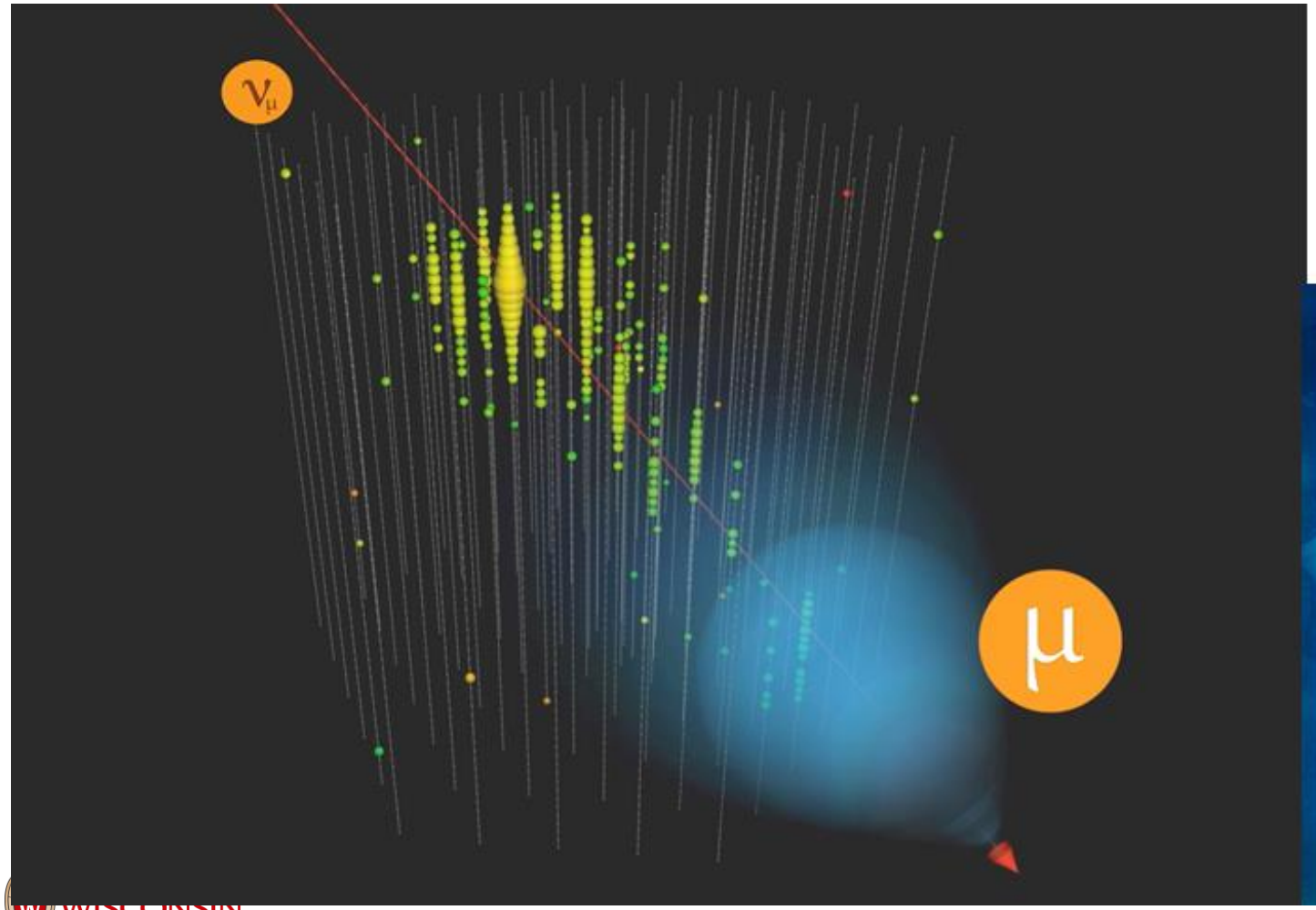
IceCube Computing

- Global heterogeneous resources pool
- Mostly shared and opportunistic resources
- Atypical resources requirements and software stack
  - Accelerators (GPUs)
  - Broad physics reach with high uptime- Lots to simulate
  - “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)
- Significant changes of requirements over the course of experiment - Accelerators, Multi-messenger Astrophysics, alerting, etc.

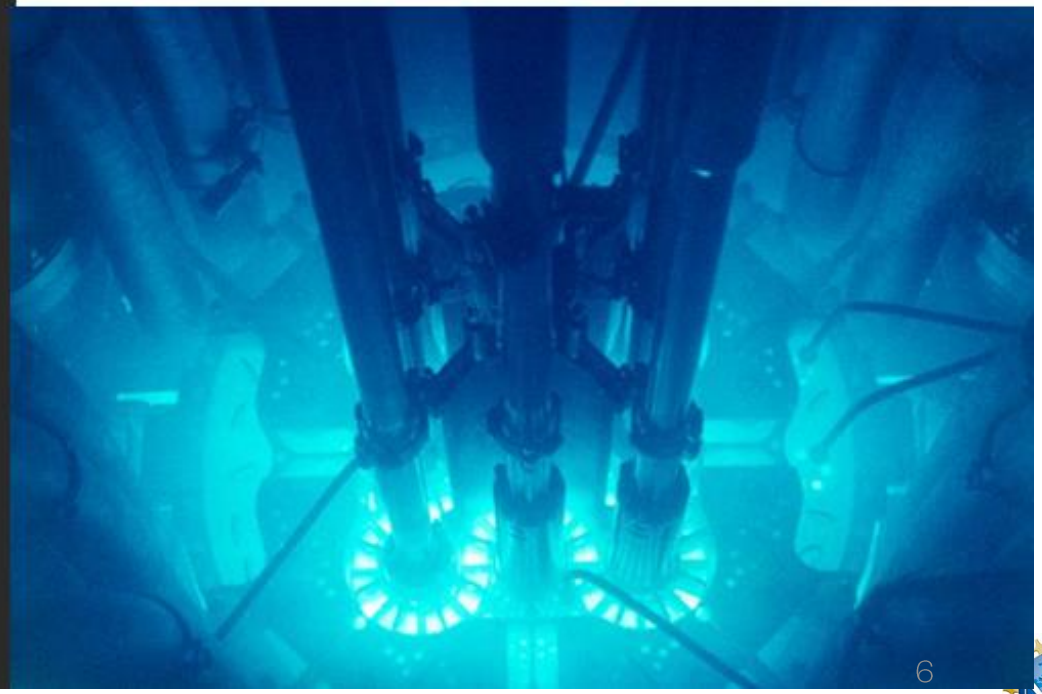




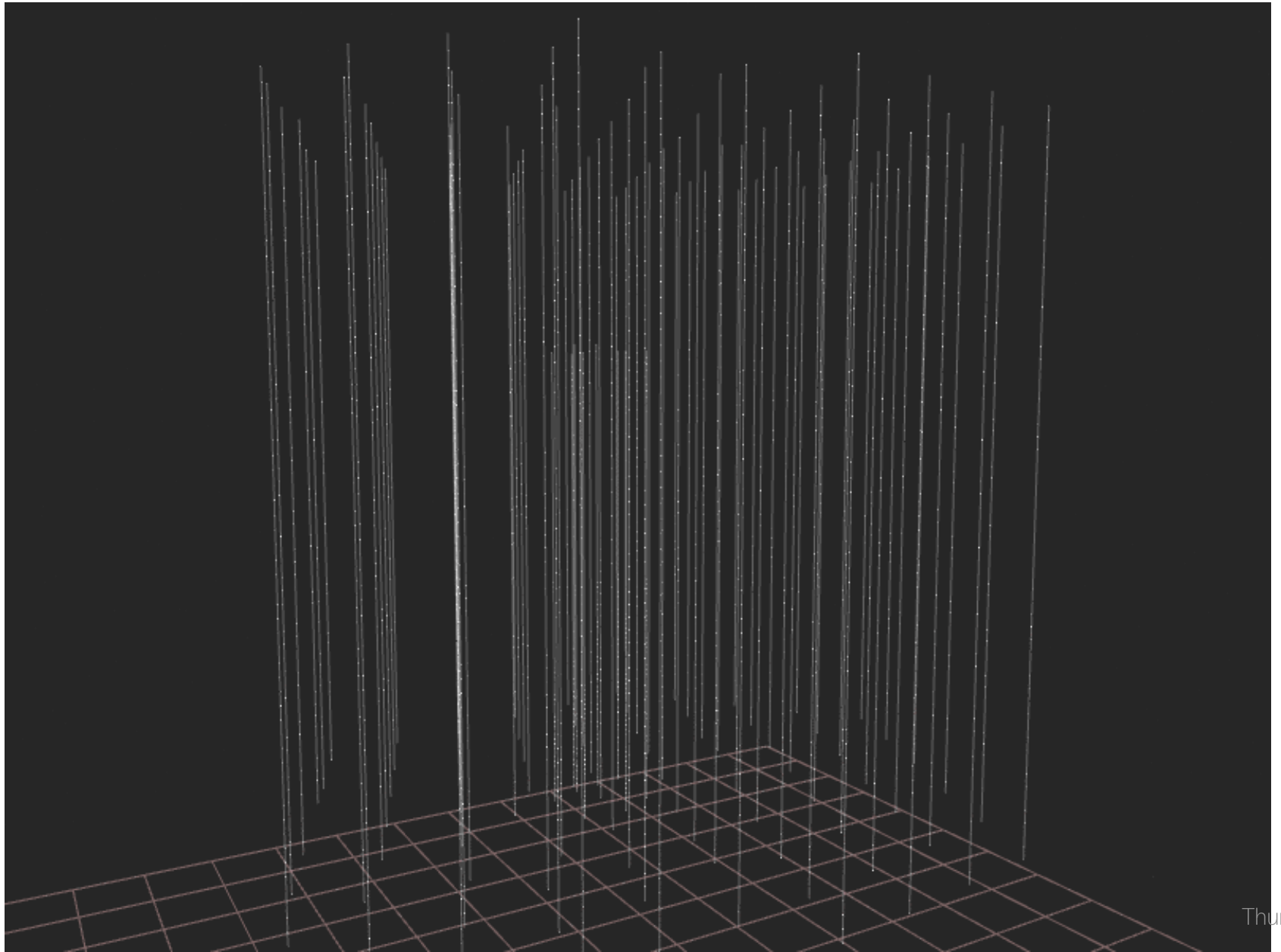
How does it work?



- Cherenkov light - Sonic boom with light
- Cherenkov light appears when a charged particle travels through matter faster than light can

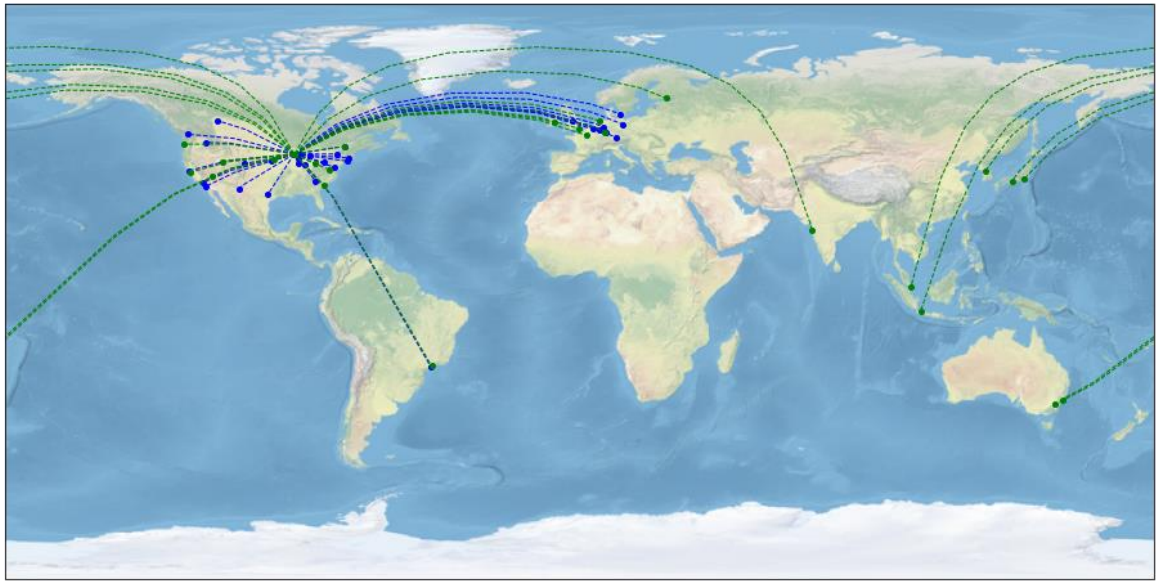


Why GPUs?

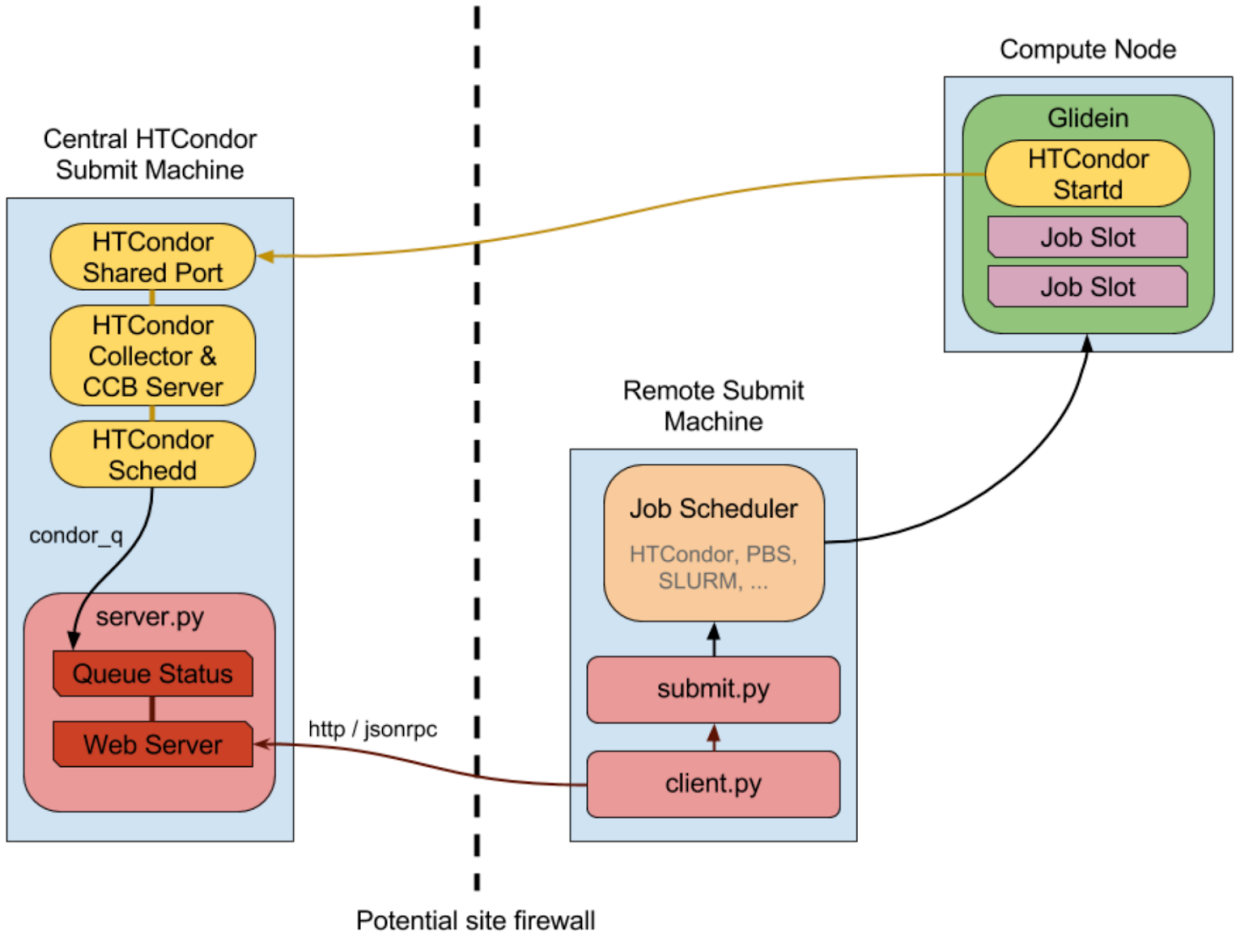


IceCube Grid

- Back in 2016…
- Grid resources, “pledged” resources (HPC allocations), and everything in between
  - Not every place has a CE
  - Integrating new resources was an issue
- Experiencing scaling issues with workflow management system
  - Workflow management system needed expert knowledge to deploy and maintain
  - >3,000 cores was hard/impossible
- HostedCE wasn’t a thing

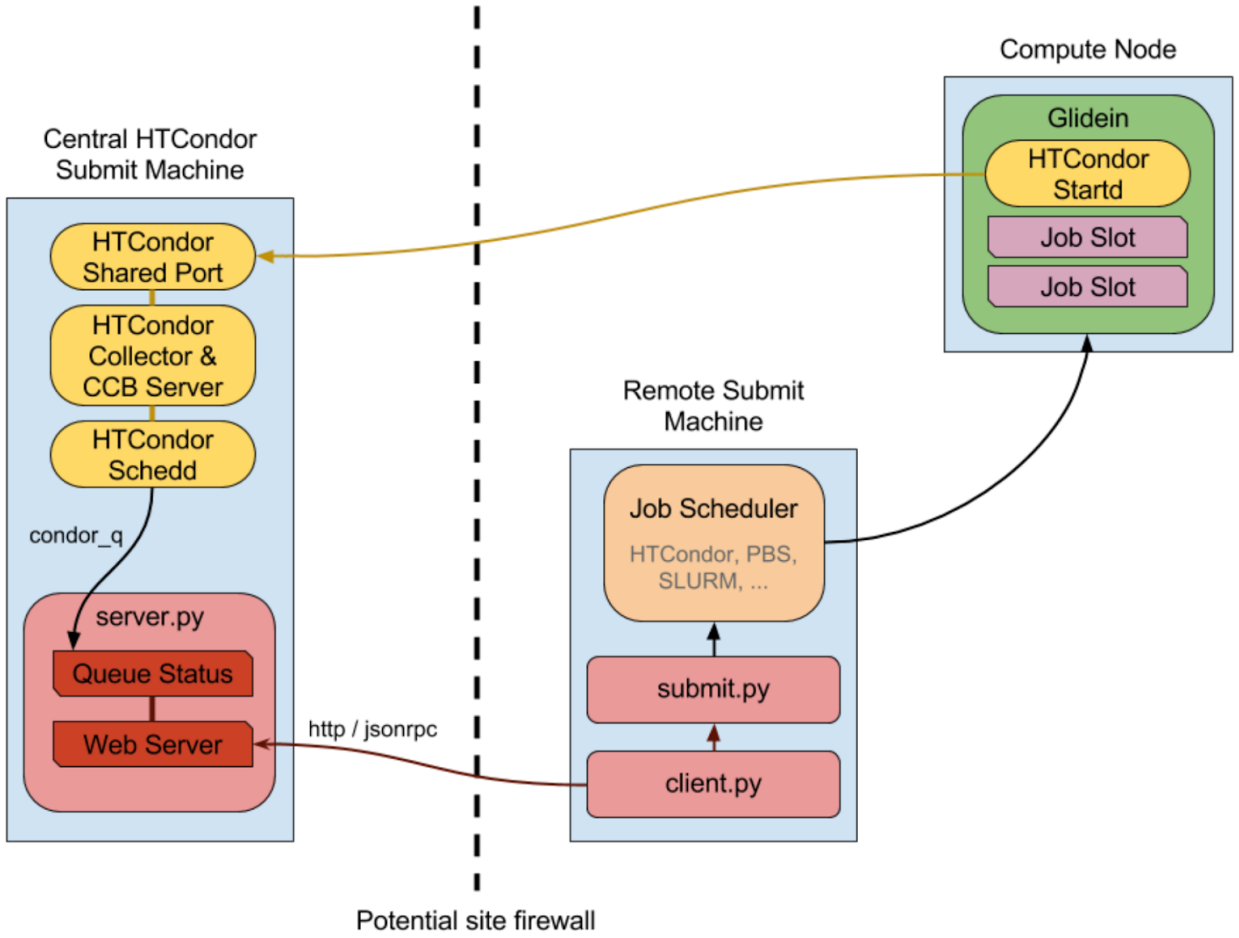


IceCube Grid – PyGlidein - I



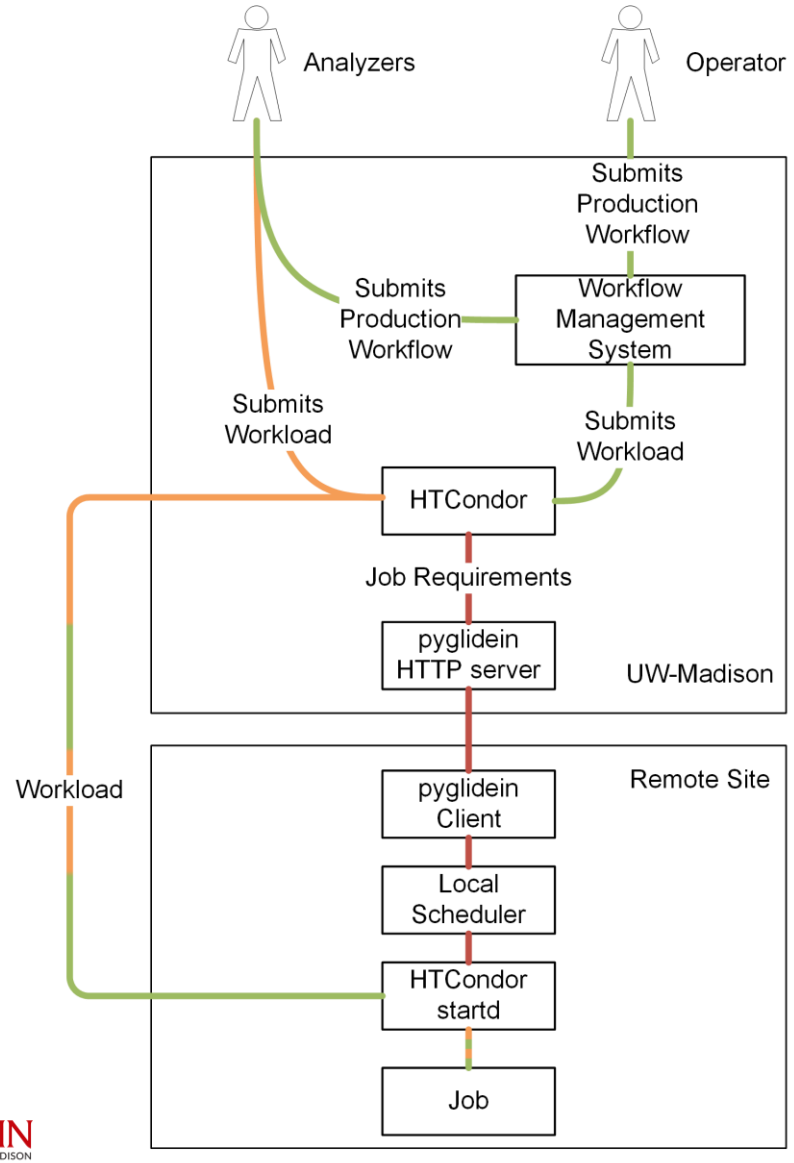
- Separate job submission from workflow management
- Lightweight design as possible
- Only difference between sites is a config file
- HTCondor checked nearly all the boxes
  - ✓ Scaling to O(100,000) cores
  - ✓ GPU and CPU support out of the box
  - × CE is too heavy – HostedCE wasn't a thing
    - × Needs it's own node/container/something
    - × GWMS factory has fixed slot sizes per resource – Would be good to be adjustable
- Why separate system?
  - Performance issues – Maximum ~3500 jobs
  - Experts needed for deployment, operation, and monitoring
  - Individual users could not use distributed resources

IceCube Grid – PyGlidein - II



- Borrowed from CHTC/OSG architecture
- Exposed the job resource requirements – CPU, Memory, Disk, GPU – via HTTP as a JSON document
- Remote client queries for job requirements and submit `startd` jobs accordingly within local resource constraints
- When `startd` executes connects back to central pool with pool password right now
- On OSGConnect this means we are running a glidein inside a glidein – Turtles all the way down
- Multiple jobs are submitted per single job in pool – Assuming other jobs will be able to use slots, otherwise dies within configurable amount of time

IceCube Grid – PyGlidein - III

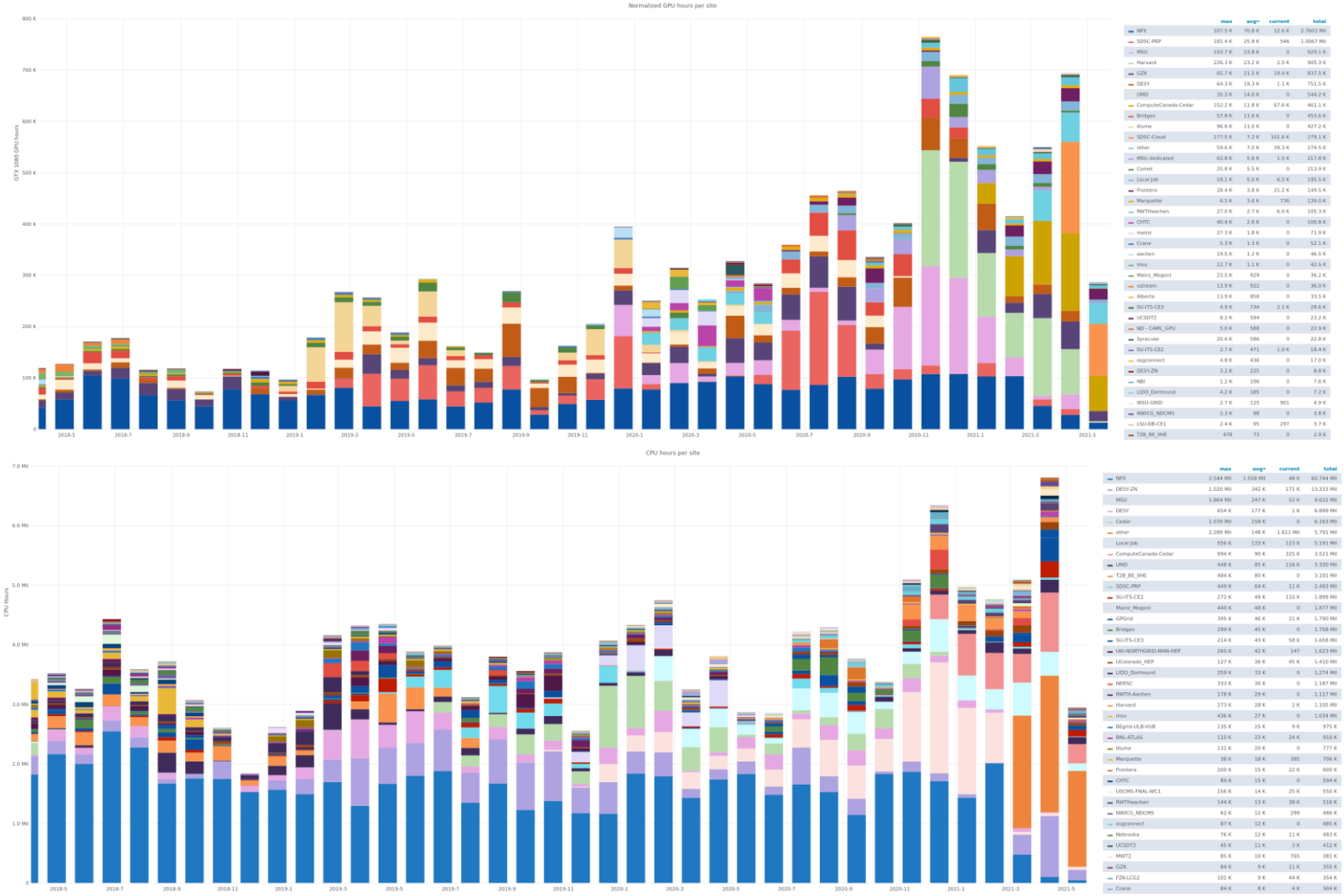


- User perspective
  - HTCondor + Data Management
  - "Just an HTCondor pool"
- Operator perspective
  - Little overhead to add cluster to pool – O(1) day depending on cluster
  - Fairly easy to monitor, e.g. `condor_status`
  - No need for a CE - Use `SSH` or `cron` for submission
  - Local container support
- Future improvements
  - Code needs clean-up – Organic growth to support multiple schedulers
  - User container support

IceCube Grid – PyGlidein – IV

- In the last 5 years most external IceCube resources are reached through pyglidein – Opportunistic and OSG resources are through CHTC
- Remarkably stable for person power invested
- Let HTCondor do the heavy lifting
  - Aggregate and schedule on distributed resources
  - Scaling problems/solutions have been established by other users
- Feature stack moves with HTCondor
  - GPU support
  - Resource splitting
  - 3rd party data copy
  - IDTokens
- Biggest issues usually relate to site rather than HTCondor
  - Network connectivity
  - Availability of CVMFS – Thanks to Dave Dykstra and `cvmfsexec` this has been solved with containers
  - No `cron` and MFA is our hardest hurdle

Usage Statistics



- Cluster can scale elastically without significant investment
- ~7 MCPU-hours and ~700 GPU-hours per month

Future Plans

- Replace our glidein with OSG glidein pilot container
- Move to IDTokens for pool authentication
- Better HTTP server
- User container support
  - Expose new classad in HTTP server – Similar to `+SingularityImage`
  - Issue is having a container that is compatible with running glidein (hard to require a certain base container) or system allowing running container inside container

Thank you!

Questions?

GPU Cloudburst Experiments

- Original Goal: Create an ExaFLOP compute pool in the cloud (80,000 NVIDIA V100) and address review panel recommendations
- Cloud provider(s) do not have those resources available – We were promised they do
  - Pre-allocated resources
  - Single cloud provider does not have those resources
- First Experiment – On Nov 16 2019 we bought all GPU capacity that was for sale in Amazon Web Services, Microsoft Azure, and Google Cloud Platform worldwide - Creating The Largest GPU Cloud Pool in History
  - 51k NVIDIA GPUs in the Cloud
  - 380 Petaflops for 2 hours (90% of DOE's Summit, No. 1 in Top 500)
  - Distributed across, US, EU, and Asia-Pacific
  - Cost: $50-150k (under NDA)
- Second Experiment – More realistic test
  - Most cost-efficient GPUs for 8 hours
  - Achieve 1 ExaFLOP-hour of compute
  - Distributed across, US, EU, and Asia-Pacific
  - Cost: ~$60k
- Third Experimenting
  - Have OSG handle the HTCondor setup, etc. and use GWMS
  - Successfully completed
  - Used T4 instances for cost effectiveness