Distributed Computing and some other Computing Thoughts

Benedikt Riedel
UW-Madison

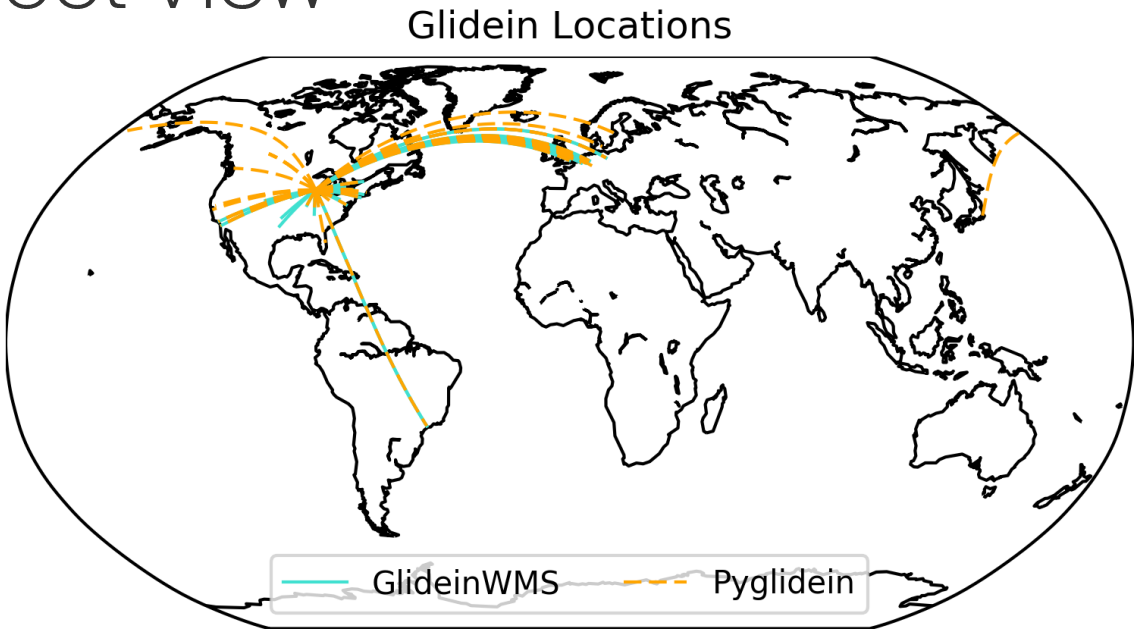
Bootcamp
13 June 2022

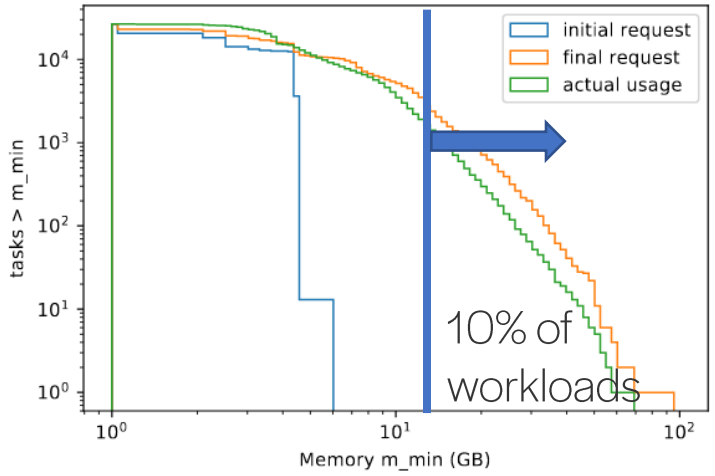
IceCube Computing – 30,000 Foot View

- 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.

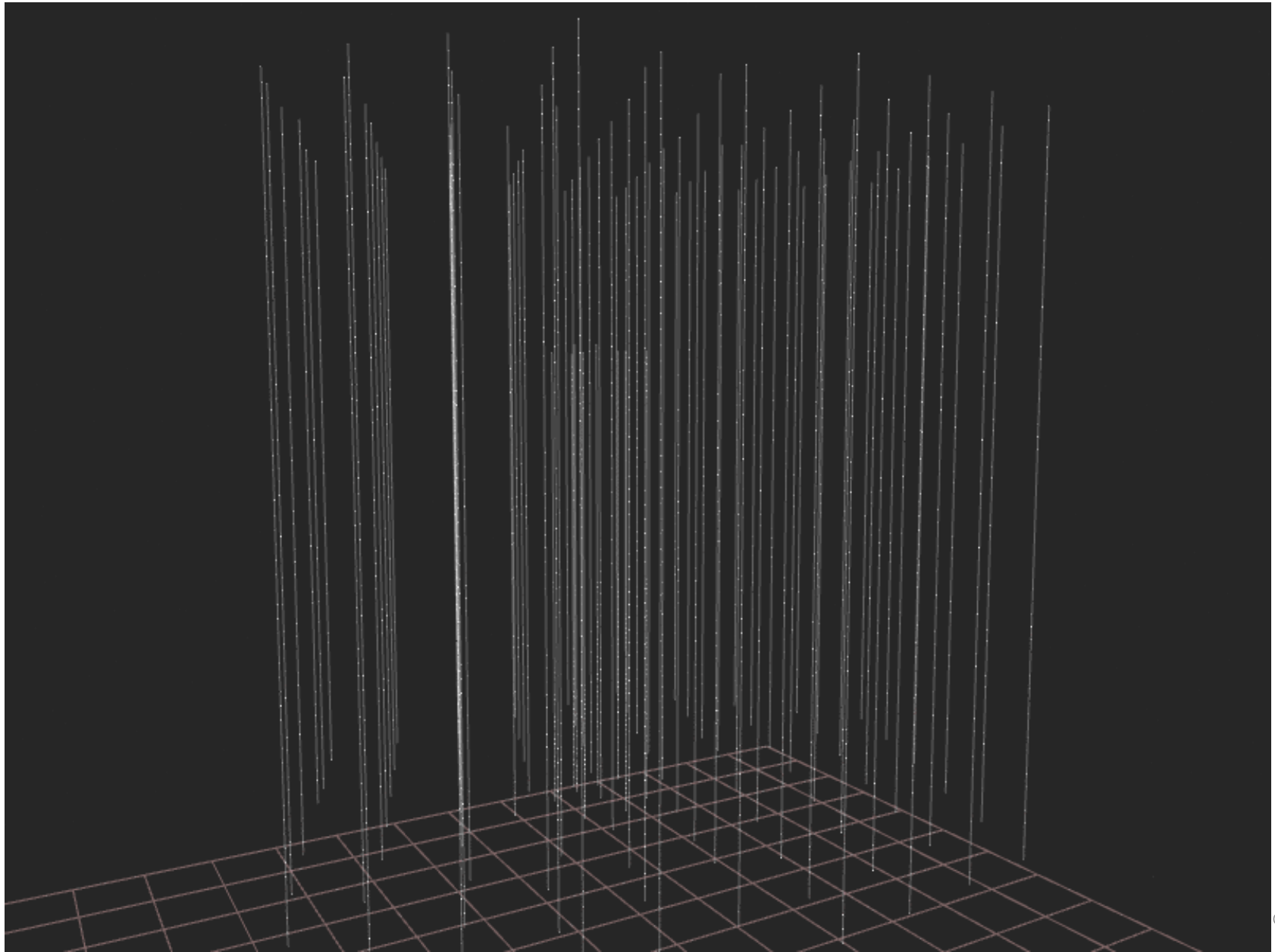
IceCube Computing – 10,000 Foot View

- 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.

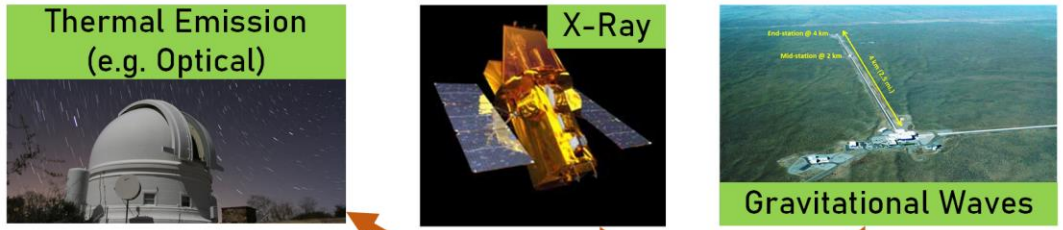


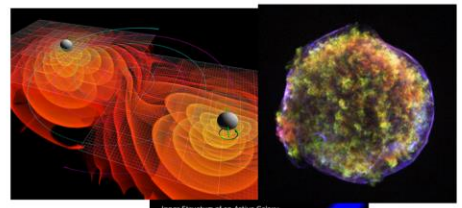


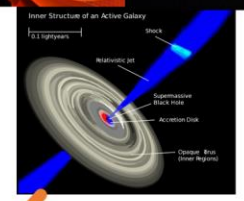
Why GPUs?



Data Transfer – High Level







UW-Madison

Raw Data

Iridium

TDRSS

Ship

Alert Data

Reduced Data

Disk

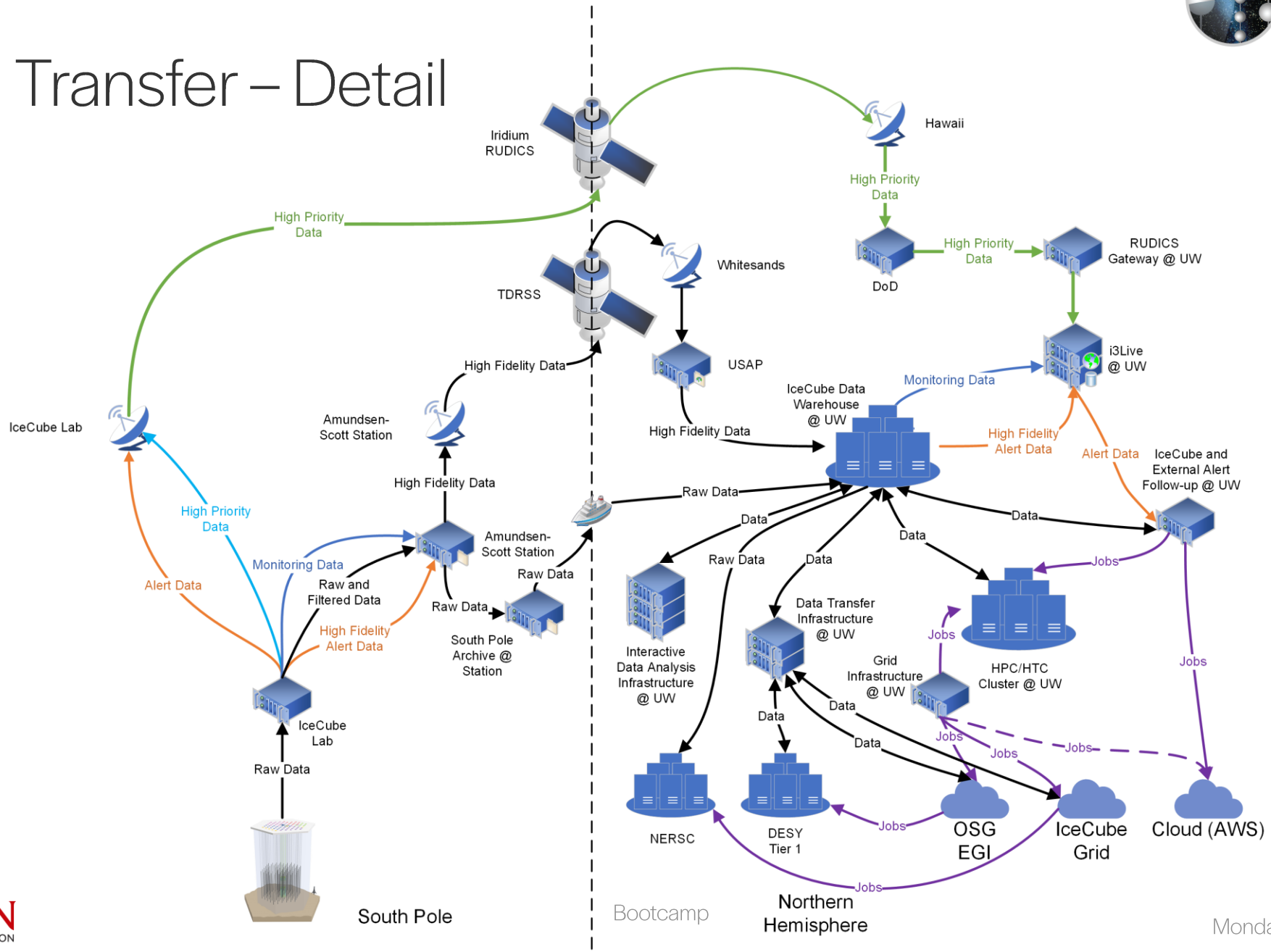
Raw Data

Raw Data

ν

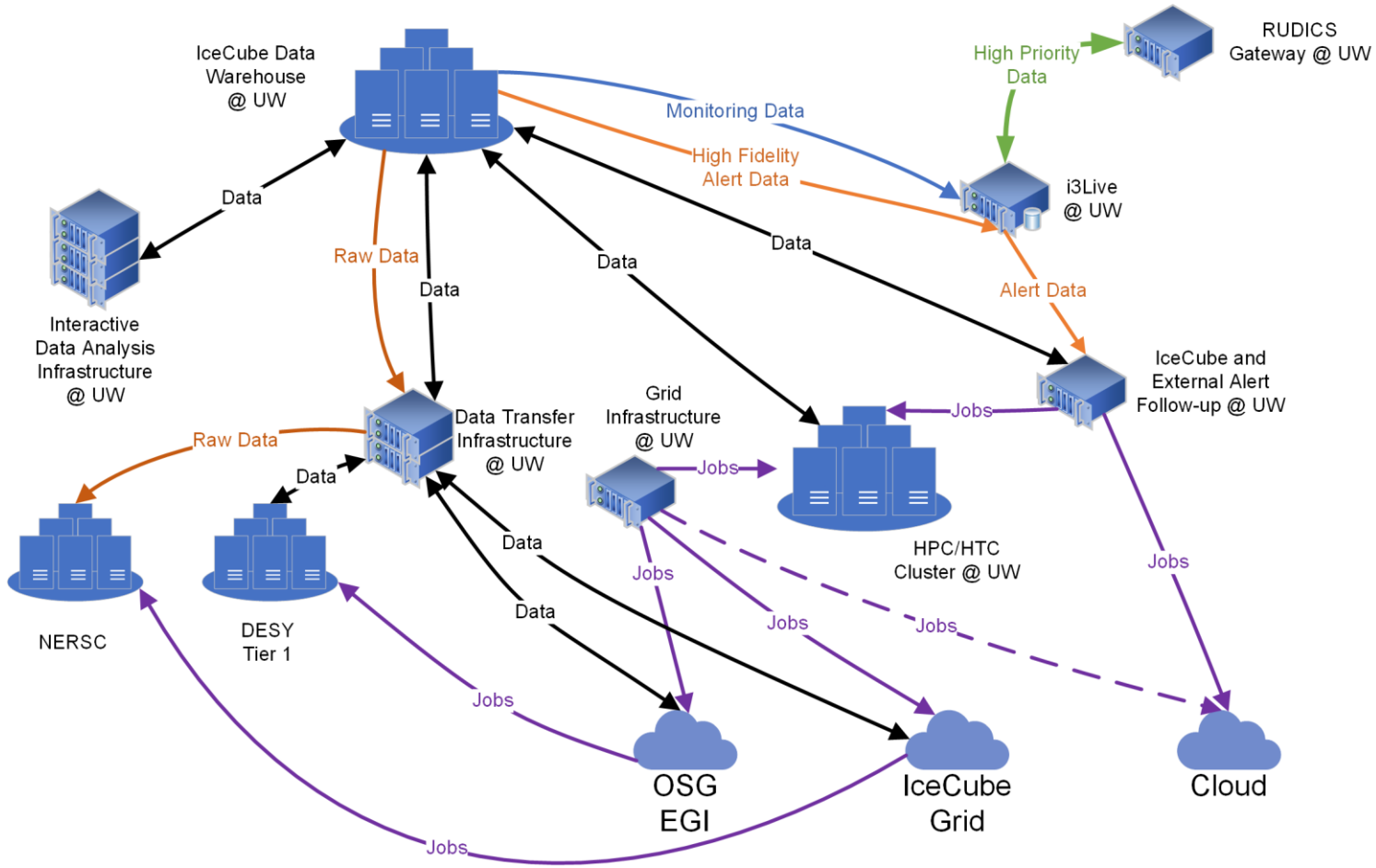
E-CAS Workshop

Data Transfer – Detail



Computing Resources

- A large portion of resources are not in Madison
- Need to aggregate resources to have sufficient resources for IceCube's science needs
  - Monte Carlo Simulation
  - Special hardware for ML/AI



Local HTCondor Cluster

- IceCube has an HTCondor computer cluster referred to as NPX located at the UW-Madison computing computing center
- Condor is the software that manages which job gets sent to which computer: htcondor.readthedocs.io
- ~7600 HT CPU cores
- ~400 GPUs
- Use submit.icecube.wisc.edu to submit your jobs to NPX

```
yourlaptop ~ $ ssh pub.icecube.wisc.edu
pub1 ~ $ ssh submit.icecube.wisc.edu
```

Local HTCondor Cluster

```
submitter ~ $ pwd
/home/briedel

submitter ~ $ ls /data/
ana  exp  sim  user  wipac

submitter ~ $ ls /cvmfs/icecube.opensciencegrid.org/
buildall.sh  data  distrib  iceprod  py2-v1  py2-v2  py2-v3
py2-v3_early_access  README  setup.sh  standard

submitter ~ $ mkdir /scratch/briedel
```

Local HTCondor Cluster

```
submitter ~ $ pwd
/home/briedel

submitter ~ $ ls /data/
ana  exp  sim  user  wipac

submitter ~ $ ls /cvmfs/icecube.opensciencegrid.org/
buildall.sh  data  distrib  iceprod  py2-v1  py2-v2  py2-v3
py2-v3_early_access  README  setup.sh  standard

submitter ~ $ mkdir /scratch/$USER
```

Local HTCondor Cluster

- Use the provided scratch space.
- Logfiles in network filesystems (/home, /data/user) can generate instability in Condor.
- keep your job logfiles in local disk (typically /scratch)

```
submitter ~ $ mkdir /scratch/$USER/; cd /scratch/$USER/

submitter ~ $ wget http://icecube.wisc.edu/~gmerino/bootcamp/job.sh

submitter ~ $ cat job.sh
#!/bin/bash
printf "Start time: "; /bin/date
printf "Job is running on node: "; /bin/hostname
printf "Job running as user: "; /usr/bin/id
printf "Job is running in directory: "; /bin/pwd
echo "Working hard..."
sleep $1
echo "Job complete!”

submitter ~ $ chmod +x job.sh
submitter ~ $ ./job.sh 5
```

Local HTCondor Cluster

```
submitter ~ $ cat job.sub
executable = job.sh
arguments = 10

log = job.log
output = job.out
error = job.err

request_cpus = 1
request_memory = 100MB
request_disk = 1GB
#request_gpus = 1

queue 1
```

- In order to submit a job to the cluster you need a job file
- Specify your executable and any command line arguments it requires
- Log: file created by condor to track job progress
- output/error: captures standard output and standard error
- Request the amount of CPU, memory, disk, and GPU
- Queue: keyword telling it to create 1 instance of the job

Local HTCondor Cluster

- You must have some idea of how much of each resource your job will use (if you don't know measure it)
- If you do not specify resources, the default for the cluster will be used.
- The default amount will vary a lot from cluster to cluster
- Do not rely on defaults!
- It is import to request the appropriate amount of resources for your job:
  - Too little -> Your job will be killed if you go over on any resource
  - Too much -> You will wait too long because there will be fewer "job slots" matching your requirements

Local HTCondor Cluster

To submit a job give condor your submit file:

```
submitter ~ $ condor_submit job.sub

Submitting job(s).

1 job(s) submitted to cluster 12898721.
```

To view your current running jobs:

```
submitter ~ $ condor_q

-- Schedd: sub-1.icecube.wisc.edu : <128.104.255.232:9618?... @ 06/07/18 11:43:06

To monitor submitted jobs: condor_q

OWNER   BATCH_NAME       SUBMITTED   DONE   RUN    IDLE  TOTAL JOB_IDS

gmerino ID: 101524801   6/7  11:43      _      _      1      1 101524801.0

Total for query: 1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended

Total for gmerino: 1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended

Total for all users: 1736 jobs; 0 completed, 0 removed, 1548 idle, 188 running, 0 held, 0
suspended
```

Submitting and monitoring

```
submitter ~ $ condor_q -nobatch
-- Schedd: sub-1.icecube.wisc.edu : <128.104.255.232:9618?... @ 06/07/18 11:50:11
 ID           OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
101524801.0   briedel         6/14  11:49   0+00:00:00 I  0    0.0 job.sh 10
```

Jobs can be submitted in batches

Individual jobs in a batch are identified by the number after the dot: the Process ID

`JobID = ClusterID.ProcId`

NOTE: Use **`condor_q -all`** if you want to see other user's jobs in the queue

Submitting and monitoring

```
submitter ~ $ condor_q -nobatch

-- Schedd: sub-1.icecube.wisc.edu : <128.104.255.232:9618?... @ 06/07/18 11:50:11

 ID           OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD

101524801.0   briedel         6/14  11:49   0+00:00:00 I  0    0.0 job.sh 10
```

ST = status

Most Common Job status:

- Idle "I": Job has not started yet… waiting in queue
- Running "R": job is currently running
- Completed: If the job has completed, it will not appear in condor_q
- Held "H": Stalled jobs. Something you need to fix

A job that goes on hold is interrupted (all progress is lost) and kept from running again. It remains in the queue in the "H" state.

Held Jobs removed by email

```
From: root <root@sub-1.icecube.wisc.edu>
Date: 14 June 2016 at 04:17
Subject: [htcondor] sub-1: held jobs removed
To:
12925663.0   briedel        6/13 23:54 Policy violation. Memory limit exceeded: 2004 MB resident > 2000 MB requested. (by user condor)
12925674.0   briedel        6/13 23:54 Policy violation. Memory limit exceeded: 2003 MB resident > 2000 MB requested. (by user condor)
12925692.0   briedel        6/13 23:54 Policy violation. Memory limit exceeded: 2005 MB resident > 2000 MB requested. (by user condor)
```

We periodically scan for held jobs in the queue, remove them and notify users via email.

Some other typical hold reasons ...

```
12276425.0   briedel      6/3  00:05 Error from slot1@e201.chtc.wisc.edu: Job failed to complete in 72 hrs
12015071.5   briedel         3/23 14:33 Error from glidein_7164_75405897@a0437: STARTER at 10.80.2.181
failed to send file(s) to <128.104.255.232:59904>: error reading from
/home/icecu038/home_cream_966794713/CREAM966794713/glide_aXZBCc/execute/dir_34476/_condor_stdout: (errno
2) No such file or directory; SHADOW failed to receive file(s) from <134.93.174.12:35154>
```

Log Files

```
submitter ~ $ cat job.log
000 (12898721.000.000) 06/13 21:50:14 Job submitted from host:
<128.104.255.232:58276?addrs=128.104.255.232-58276>
...
001 (12898721.000.000) 06/13 21:52:33 Job executing on host:
<144.92.166.137:27680?addrs=144.92.166.137-27680>
...
006 (12898721.000.000) 06/13 21:52:33 Image size of job updated: 1
     0  -  MemoryUsage of job (MB)
     0  -  ResidentSetSize of job (KB)
...
005 (12898721.000.000) 06/13 21:52:43 Job terminated.
     (1) Normal termination (return value 0)
          Usr 0 00:00:00, Sys 0 00:00:00  -  Run Remote Usage
          Usr 0 00:00:00, Sys 0 00:00:00  -  Run Local Usage
          Usr 0 00:00:00, Sys 0 00:00:00  -  Total Remote Usage
          Usr 0 00:00:00, Sys 0 00:00:00  -  Total Local Usage
     314  -  Run Bytes Sent By Job
     281  -  Run Bytes Received By Job
     314  -  Total Bytes Sent By Job
     281  -  Total Bytes Received By Job
     Partitionable Resources :    Usage  Request Allocated
                    Cpus :                                 1         1
                         Disk (KB) :                  12   102400    940361
                         Memory (MB) :                 0      100       100
```

Output Files

```
submitter ~ $ cat job.out
Start time: Wed Jun 7 15:53:27 GMT 2017
Job is running on node: hibat0106.cmsaf.mit.edu
Job running as user: uid=10125(osg01) gid=10125(osg01) groups=10125(osg01),10005(osg)
Job is running in directory:
/export/data1/condor/execute/dir_144460/glide_Rvbozm/execute/dir_111261/glidein/execute.18.12.6.106-112180/di
r_122685

Working hard...

Job complete!
```

Finding Job Attributes

```
sub-1 ~ $ condor_q -long 12887688.0

JobStatus = 2

LastJobStatus = 1

User = “gmerino@icecube.wisc.edu”

Err = "/scratch/gmerino/Data/2013/logs/12887688.err"

Out = "/scratch/gmerino/Data/2013/logs/12887688.log" NumJobStarts = 1

Args = "-g
/data/exp/IceCube/2013/filtered/level2/0505/Run00122300/Level2_IC86.2013_data_
Run00 122300_0505_0_9_GCD.i3.gz -i
/data/exp/IceCube/2013/filtered/level2/0505/Run00122300/Level2_IC86.2013_data_
Run00 122300_Subrun00000070.i3.bz2 -o
/data/ana/Cscd/StartingEvents/exp/IC86_2013/burnsample/l3/00122300/Level2_IC86
.2013 _data_Run00122300_Part00000070.i3.bz2”

RemoteHost = “slot1@e281.chtc.wisc.edu"

ResidentSetSize_RAW = 1200308

DiskUsage_RAW = 891690

RemoteUserCpu = 7669.0
```

Useful Job Attributes

JobStatus: number indicating Idle (1), Running (2), Held (5), etc

RemoteHost: where the job is running

ResidentSetSize_RAW: Maximum observed physical memory in use by the job in KiB while running.

DiskUsage_RAW: Maximum observed physical memory in use by the job in KiB while running.

RemoteUserCpu: The total number of seconds of user CPU time the job has used.
EnteredCurrentStatus: time of last status change

NumJobStarts: number of times the job started executing

https://htcondor.readthedocs.io/en/latest/classad-attributes/job-classad-attributes.html

Displaying Job Attributes

Use the "-autoformat" option for condor_q

```
sub-1 $ condor_q -af JobStatus ClusterID ProcId RemoteHost ResidentSetSize_RAW
2 12892531 0 slot1@glidein_235900_283164684@cabinet-0-0-7.t2.ucsd.edu 1272208
2 12892986 0 glidein_10345_623188368@jux7c.zeuthen.desy.de 1290364
2 12893002 0 slot1@e137.chtc.wisc.edu 1181296
```

The "-constraint" option can also be handy

```
sub-1 $ condor_q -c jobstatus==2 -af JobStatus ClusterID ProcId RemoteHost
ResidentSetSize_RAW
2 12892531 0 slot1@glidein_235900_283164684@cabinet-0-0-7.t2.ucsd.edu 1272208
2 12892986 0 glidein_10345_623188368@jux7c.zeuthen.desy.de 1290364
2 12893002 0 slot1@e137.chtc.wisc.edu 1181296
```

Displaying Machine Attributes

```
sub-1 $ condor_status -long -limit 1
...
Cpus = 4
Gpus = 0
Memory = 16000
...
GLIDEIN_Site = "UMD"
OpSysAndVer = "Ubuntu14"
...
GLIDEIN_Max_Walltime = 172500
MonitorSelfAge = 44406
...
```

Grid Computing

IceCube has access to more computer resources at other sites besides UW-Madison, referred to as “The Grid”

- Access is provided at `sub-1.icecube.wisc.edu`
- The Grid also uses HTCondor but does not have access to the `/data/sim` and `/data/exp` directories so files must be transferred using grid certificates (soon to be changed to HTTP and part of HTCondor)
- https://wiki.icecube.wisc.edu/index.php/Condor/Grid

DAGMan

DAGMan is a tool that comes bundled with HTCondor. It can do two useful things:

- Control number of running jobs
- Handle inter-job dependencies

Example:

```
# file name: dagman.submit
JOB job1 job.condor
VARS job1 gcd="gcd.i3.gz"
VARS job1 infilename="input.i3.bz2"
VARS job1 outfilename="output.i3.bz2"
JOB job2 job.condor
VARS job1 gcd="gcd.i3.gz"
VARS job2 infilename="input2.i3.bz2"
VARS job2 outfilename="output2.i3.bz2"
```

```
# file name: job.condor
executable = my_first_icetray_script.py
output = job.$(Cluster).out
error = job.$(Cluster).err
log = job.log
notification = never

Arguments = $(gcd) $(infilename)
$(outfilename)
queue
```

IceProd2

- IceCube Specific scheduler for the grid
- Used by simulation production to create official datasets
- Describe jobs to run using json
- Handles File transfers to data warehouse
- Uses web interface

General Advice

- Never submit anything to the cluster without first estimating the memory usage and run time
- Always write your data to the provided scratch directory: ${_CONDOR_SCRATCH_DIR}”
- Always write logs to the scratch directory (not `/data/user`)

Links

Good tutorials can be found in the HTCondor week 2018:

- An Introduction to Using HTCondor
- Introduction to Workflows with DAGMan
- HTCondor advanced job submission

Or on the user's manual:
https://htcondor.readthedocs.io/en/latest/users-manual/quick-start-guide.html

IceCube-specific information:

- Grid Computing: pages in the IceCube wiki
- Condor: pages in the IceCube wiki
- https://iceprod2.icecube.wisc.edu: Iceprod2

Contact

Email:

- Problems/questions: help@icecube.wisc.edu
- discussion/information (mailing list): icecube-computing@icecube.wisc.edu

Slack channels:

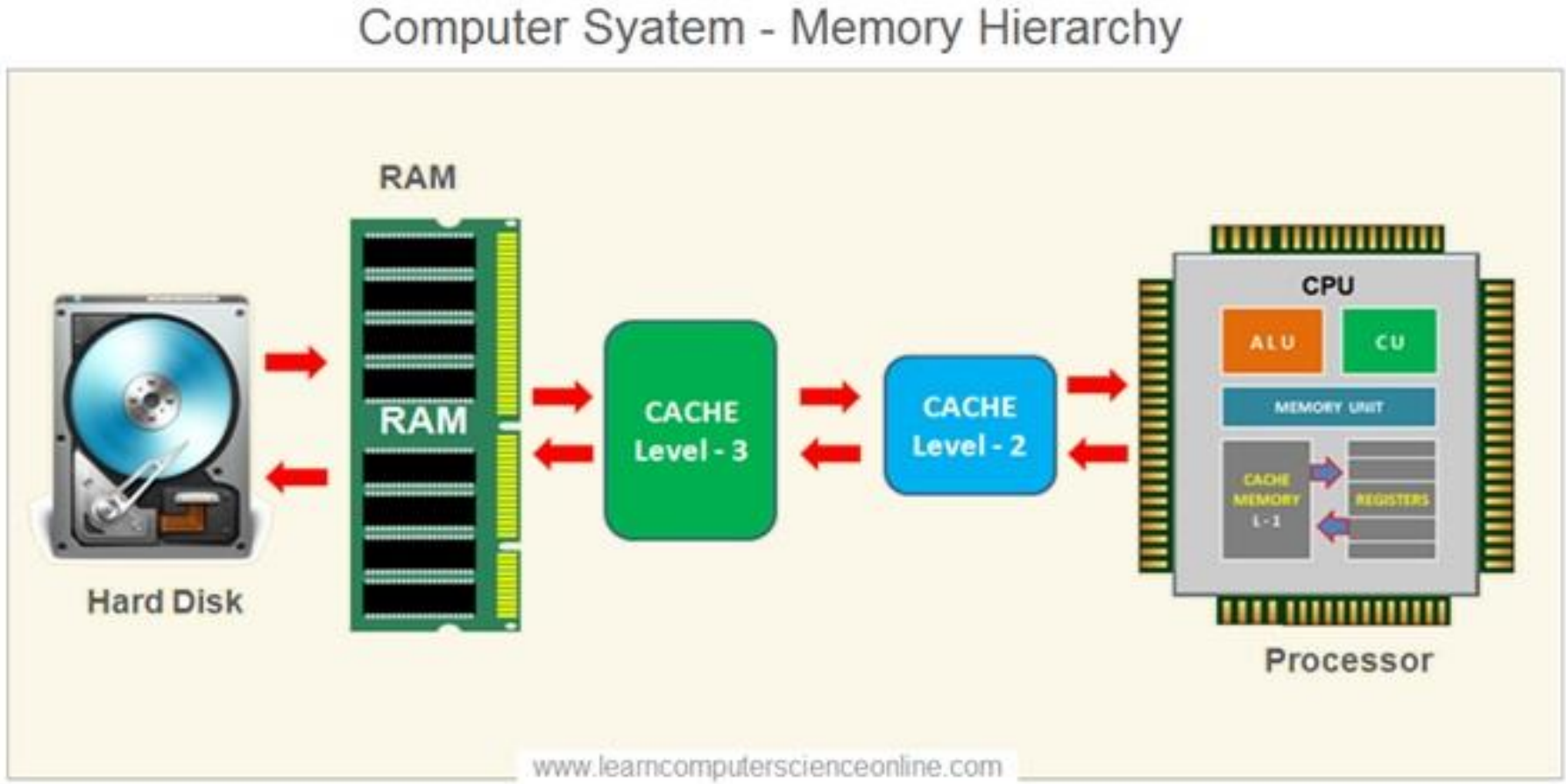
- #icecube-it - Questions about HTCondor, the grid or any computing infrastructure
- #software - Questions about icetray or any software

Now an interlude about
Computers Systems
...
Why?
...
Machine Learning/Artificial
Intelligence and GPUs

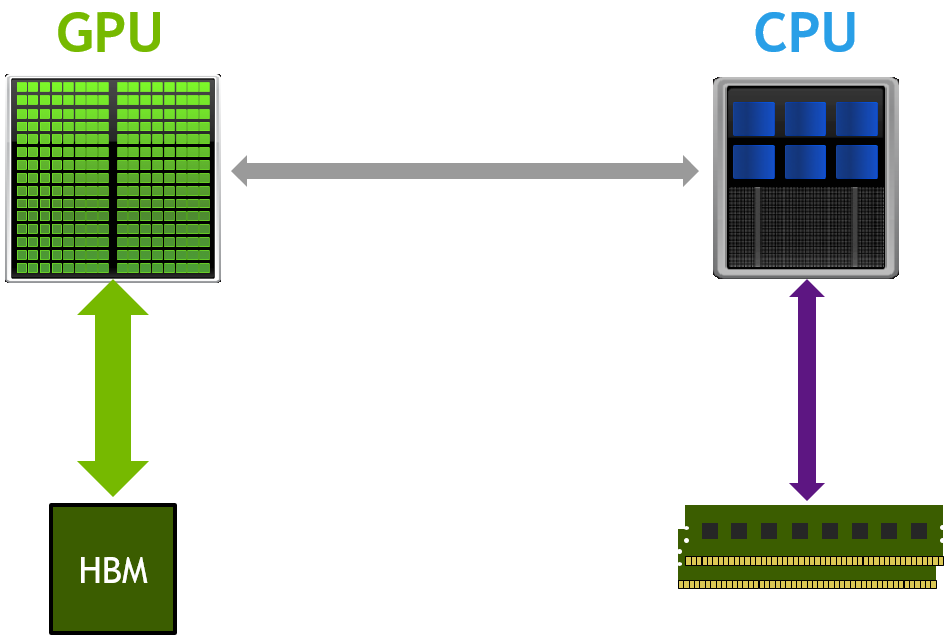
Why does this matter?

- A lot of you will use or rely on Machine Learning/Artificial intelligence
- ML/AI has made helped make scientific breakthroughs and improvements over "traditional" methods
  - Drug Discovery: https://www.nature.com/articles/s41573-019-0024-5
  - Protein Folding: https://www.nature.com/articles/d41586-021-03499-y
  - LHC: https://exatrkx.github.io/
  - IceCube: https://arxiv.org/abs/2101.11589
- What is the issue?:
  - No good definition of error
  - What is happening inside the ML/AI algorithm?
  - Changes in the computing model

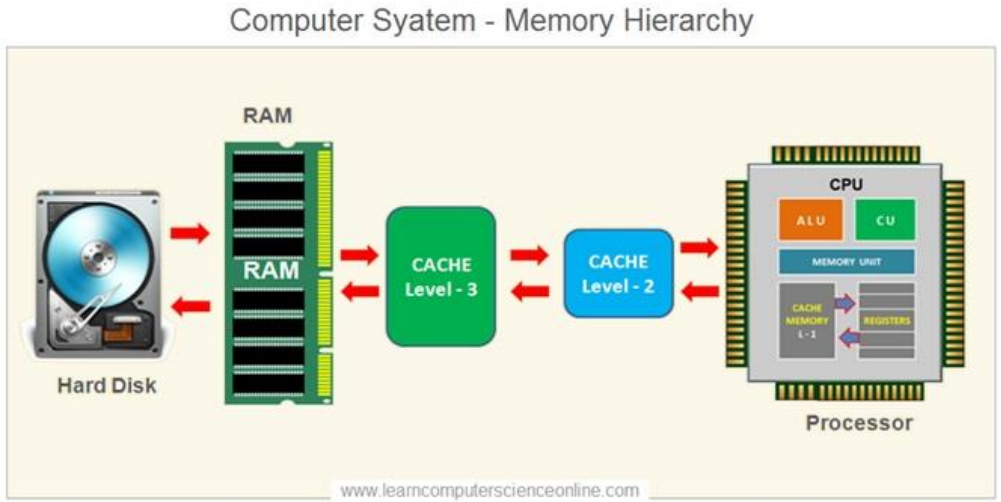
Normal Case


Computer Syatem - Memory Hierarchy
RAM
RAM
CACHE
Level - 3
CACHE
Level - 2
CPU
ALU
CU
MEMORY UNIT
CACHE
MEMORY
L-1
REGISTERS
Hard Disk
Processor
www.learncomputerscienceonline.com

ML/AI GPU Case

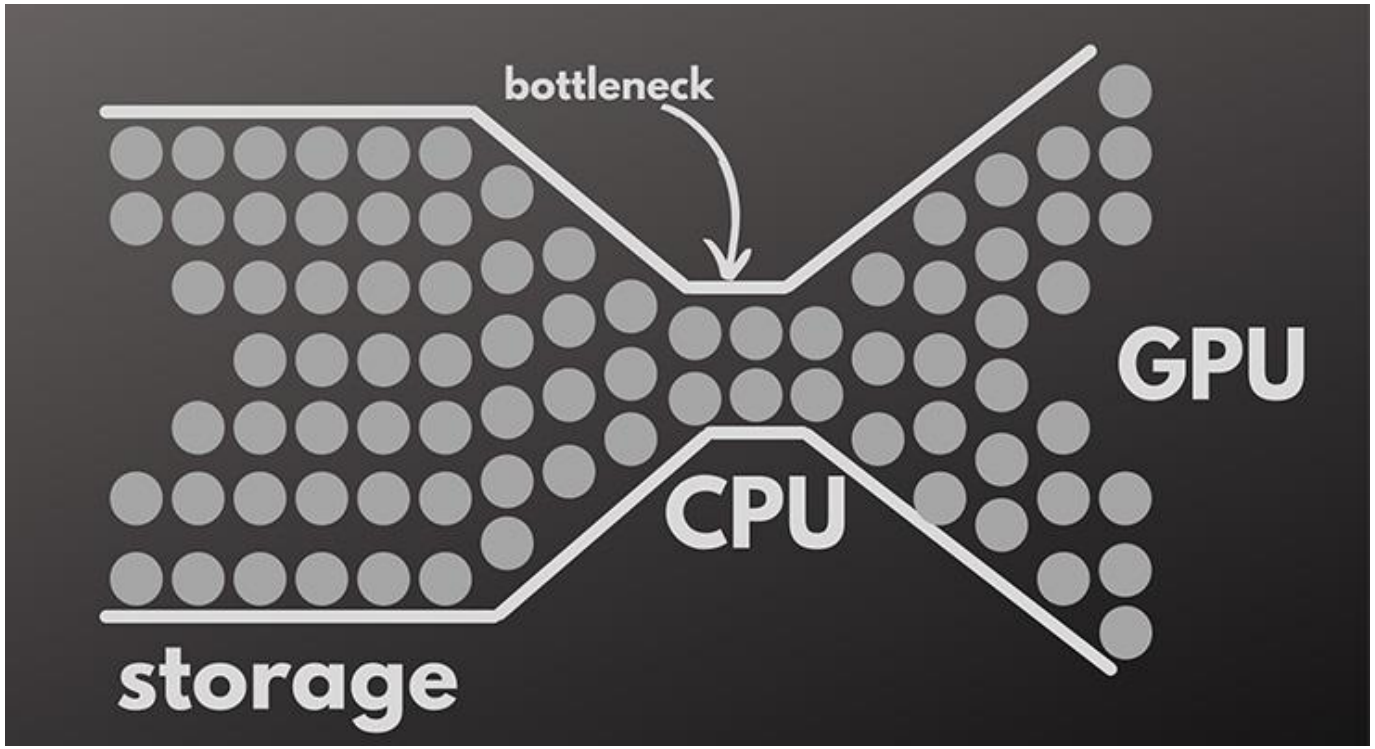


VS.



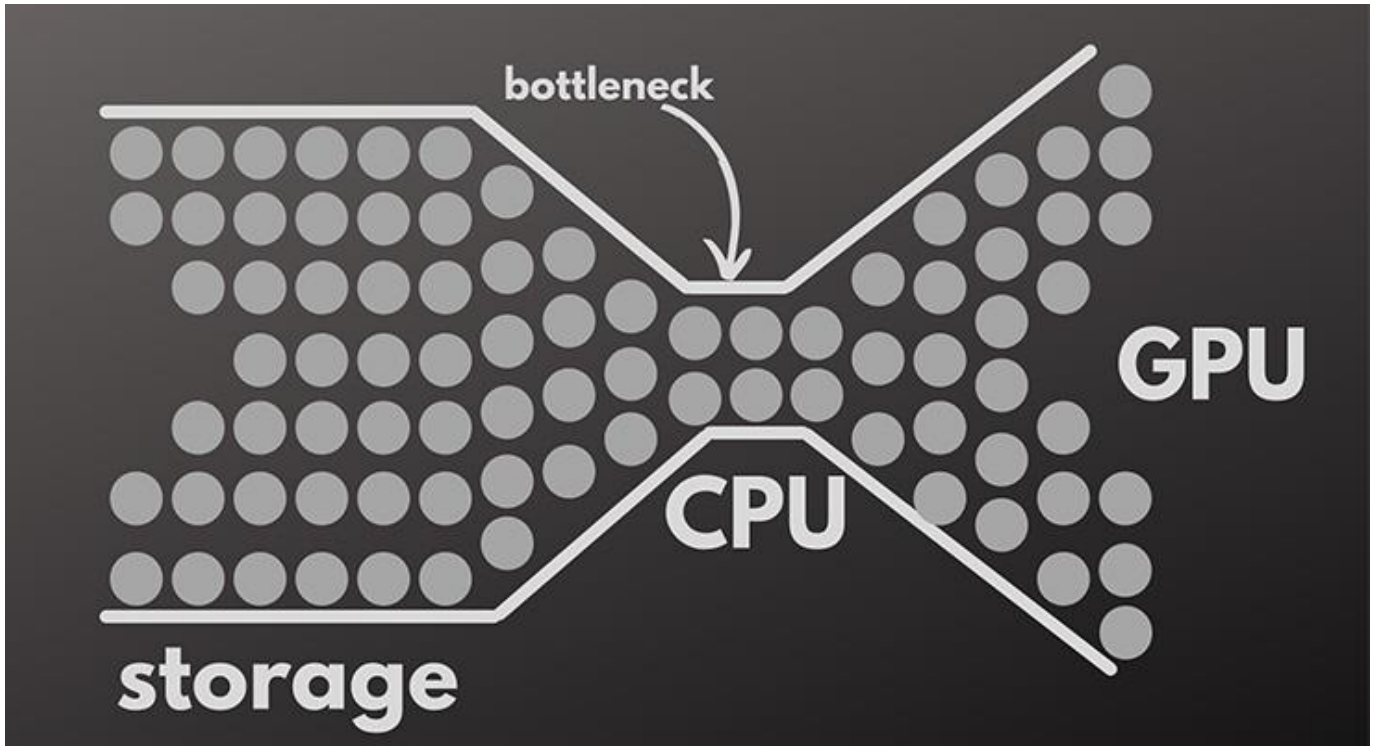
ML/AI GPU Case

- **Per event** speed-up for ML/AI is great
- But we don't just process one event…
  - When using a GPU you need to move the data through the CPU and CPU-GPU connection (PCIe) – Extra step
  - Inference (running the ML/AI algo on your data) is poor at GPU utilization
    - Why? - Data cannot be fed fast enough to the GPU



Why does it matter?

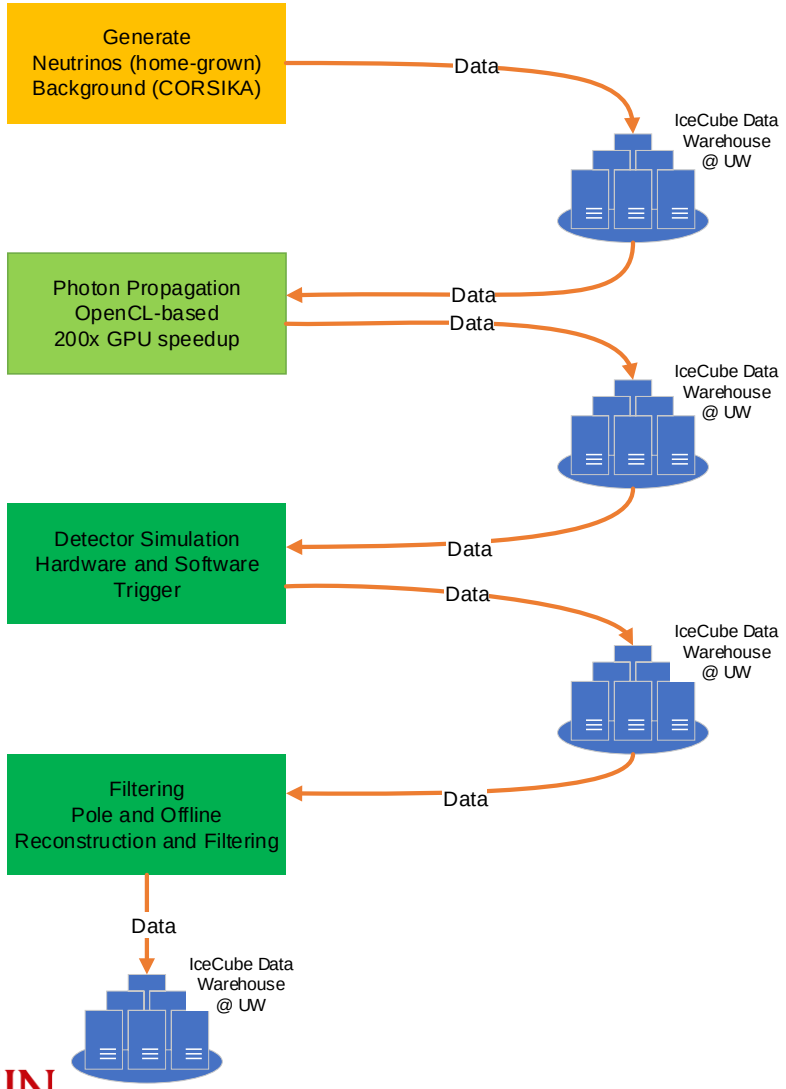
- GPUs might be 100+x faster than CPUs on a **per event basis**, but…
  - Need to consider the time it takes to push data to the GPU…
  - GPUs resources are limited compared to CPU resources – Easily 100x CPU resources



Thank you!

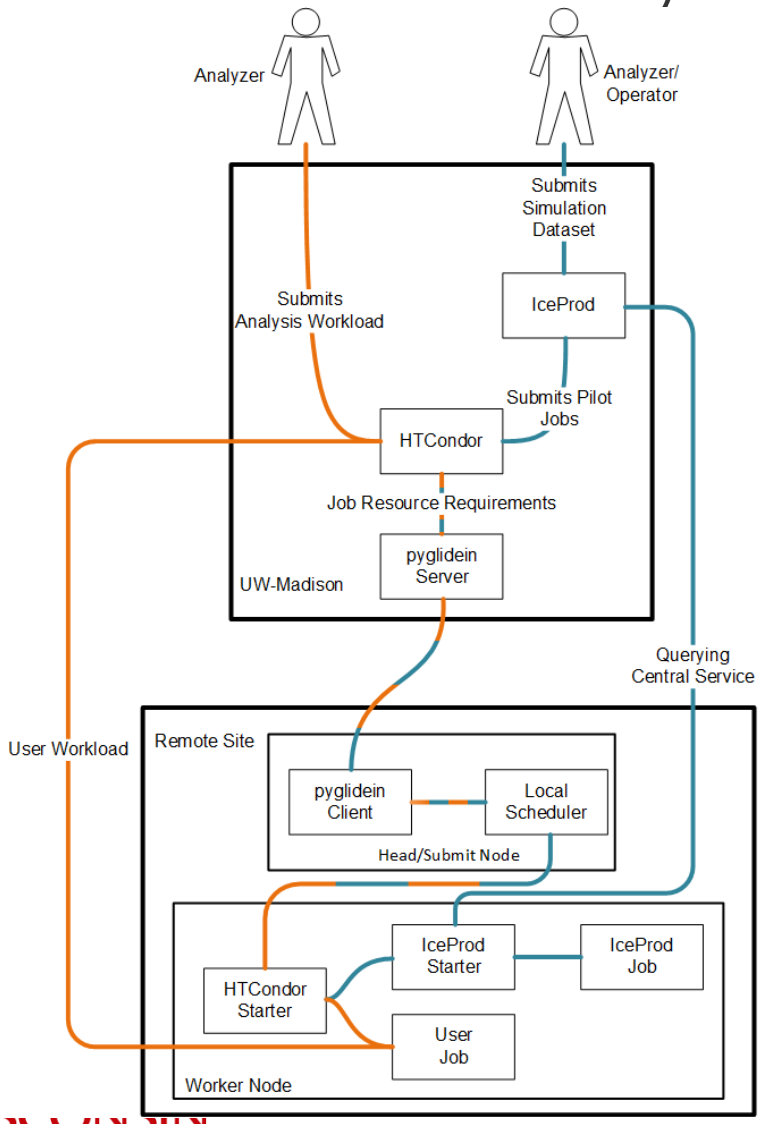
Questions?

IceCube Grid – Simulation Workflow



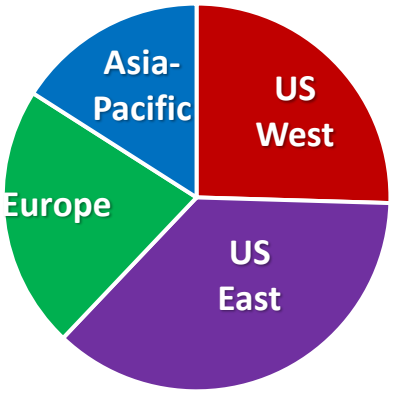
- Fairly straightforward particle physics-like workflow
- Big constraint is lack of dedicated resources
  - No data aware scheduling
  - Lots of data movement – Lots of time wasted to move data
- Different steps can have drastically different requirements

IceCube Grid – PyGlidein

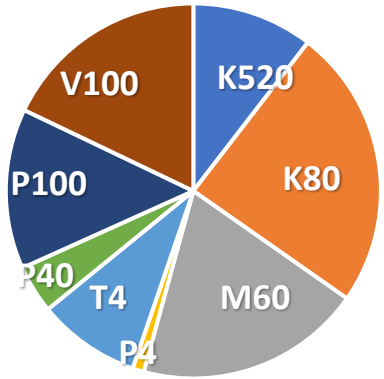


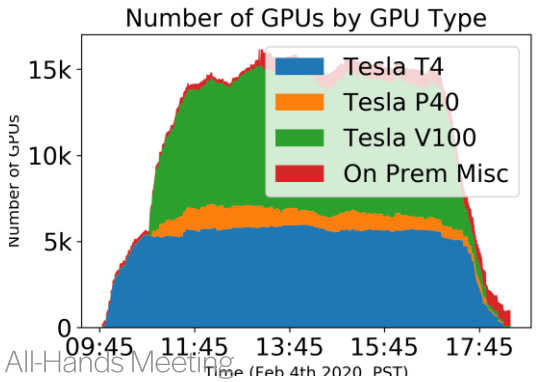
- User perspective
  - HTCondor + Data Management
  - "Just an HTCondor pool"
- Operator perspective
  - Little overhead to add cluster to pool
  - Fairly easy to monitor, e.g. `condor_status`
  - No need for a CE - Use `SSH` or `cron` for submission
  - Local container support
- Future improvements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

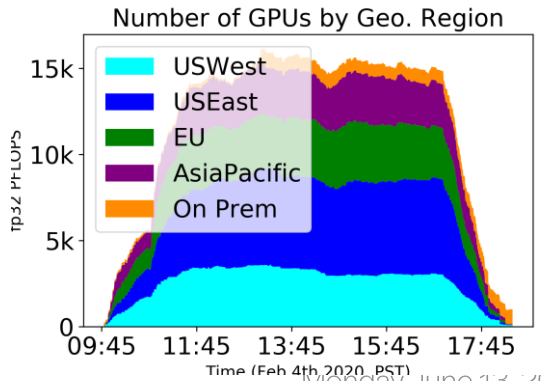
GPU Cloudburst Experiments



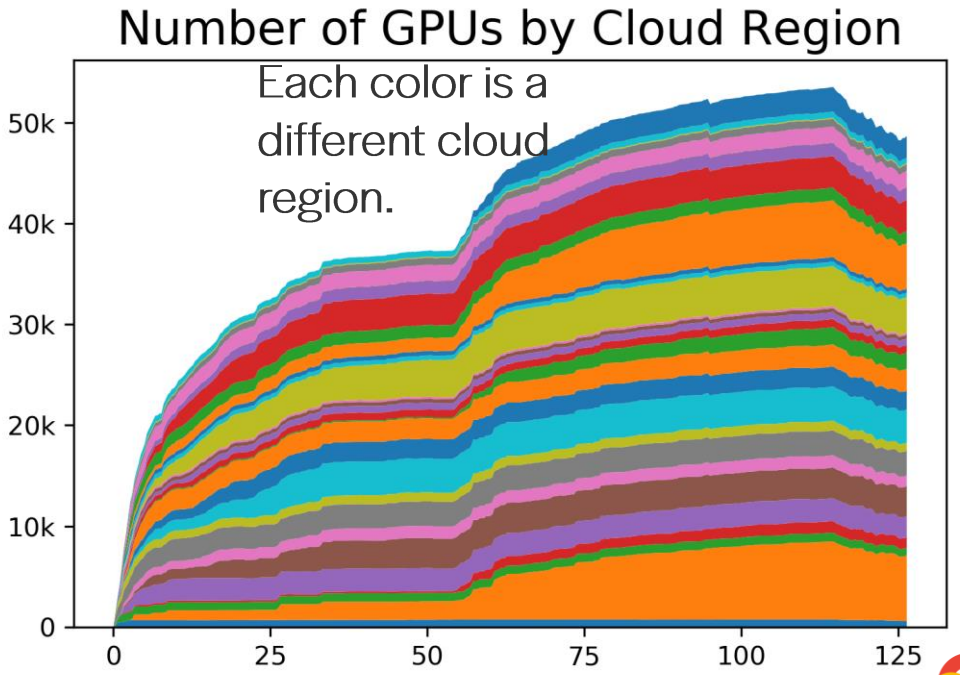
- Original Goal: Create an ExaFLOP compute pool in the cloud (80,000 NVIDIA V100) and address review panel recommendations
- Cloud provider(s) do not have those resources available – We were promised they do
  - Pre-allocated resources
  - Single cloud provider does not have those resources
- First Experiment – 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







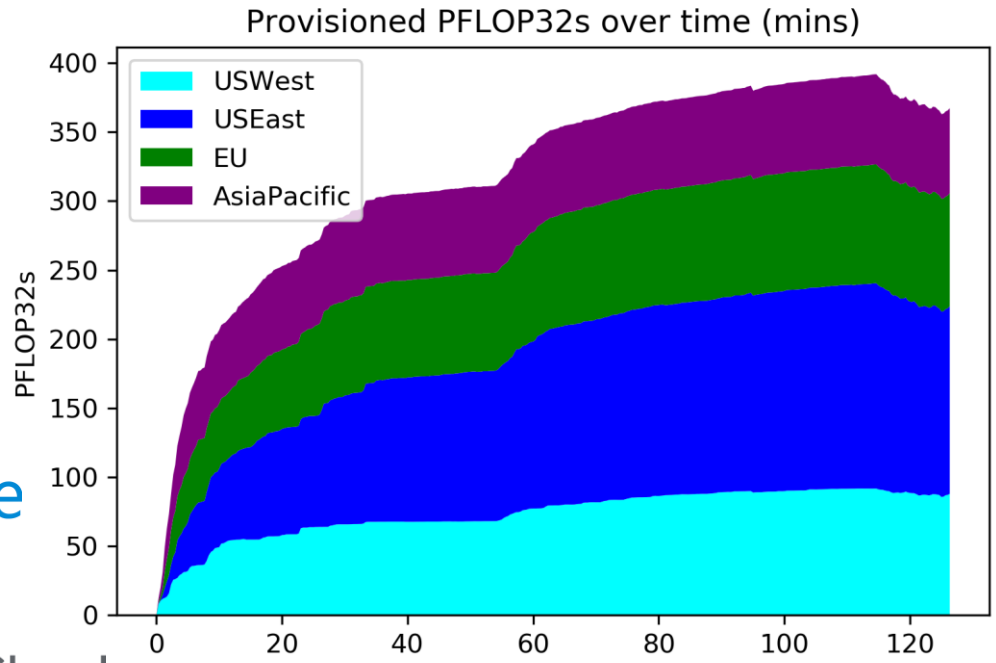
GPU Cloudburst – 1st Experiment



aws

Azure

Google Cloud

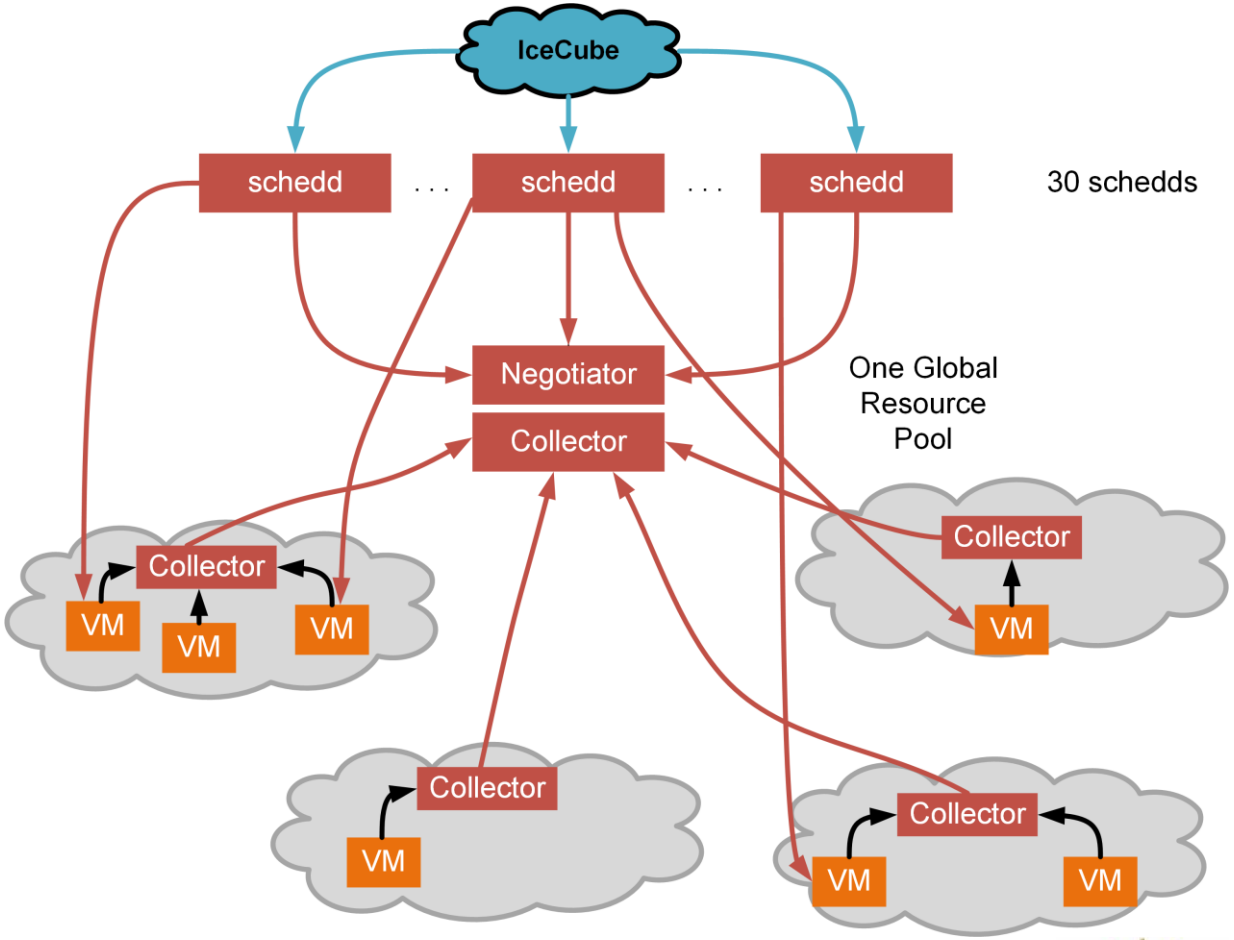


Peaked at 51,500 GPUs

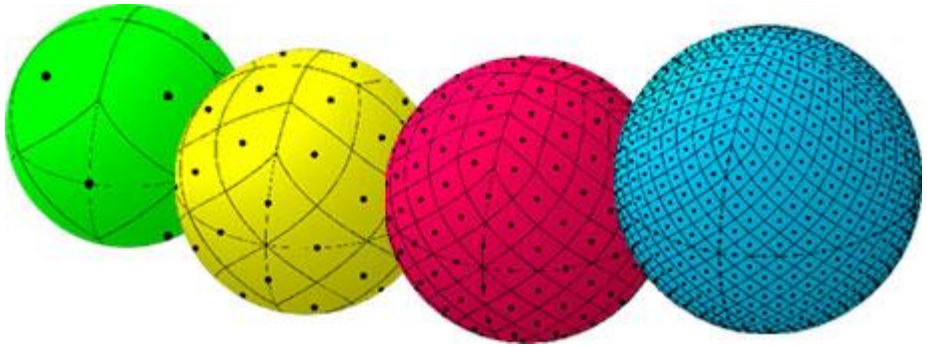
Total of 28 Regions in use.

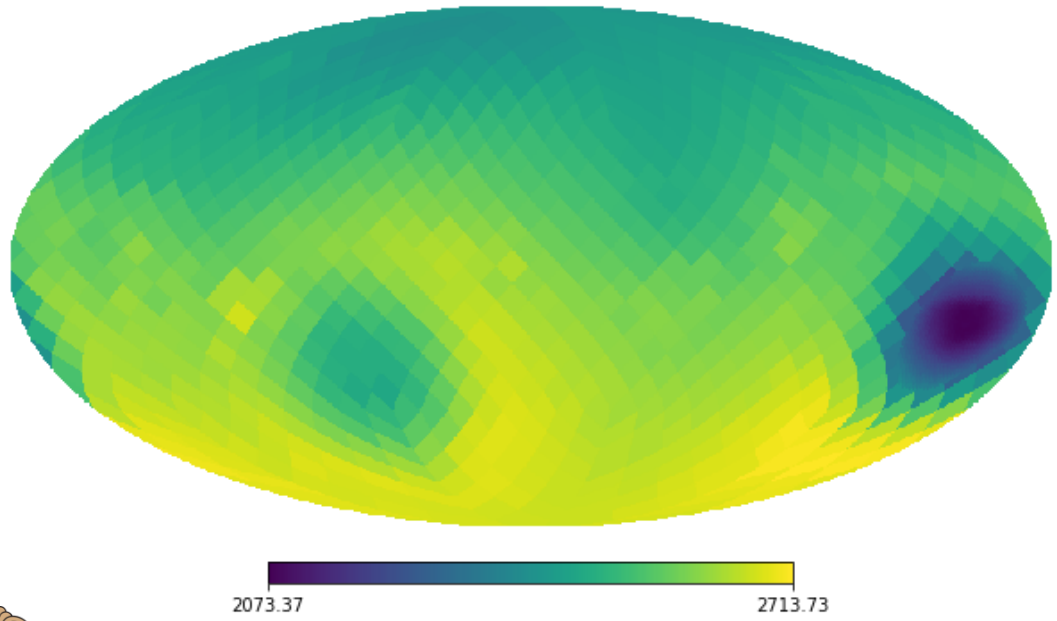
GPU Cloudburst Technology

- Multi-collector HTCondor setup – Already well-established
- Collector in each cloud region to reduce load on start-up – No idea where resources would be
- Workload is computing heavy compared to typical IceCube load – Reduce potential networking cost
- 1st demo: In and output data stored in cloud
- 2nd demo: Input came from UW, output stored in cloud



Multi-Messenger Astrophysics – Reconstruction





- Most accurate directional reconstruction comes by scanning across the sky
  - Split sky into constant surface area pieces
  - Test each directional hypothesis against likelihood
  - Create directional likelihood map
  - Gives most probable direction and error
- Each hypothesis calculation is independent – Easy to split up workload across O(1000[000]) or cores