Group (E)xcellent

Bill, Aaron, Trevor, Jade, Alex G, Elliott

```
Name                OpSys      Arch   State     Activity LoadAv Mem     ActvtyTi

slot1@awang         LINUX      X86_64 Unclaimed Idle      0.000 515685  0+23:59:
slot1@goodwill      LINUX      X86_64 Unclaimed Idle      0.000 515685  2+06:01:
slot1@jchismar      LINUX      X86_64 Unclaimed Idle      0.000 515685  2+06:07:

                     Total Owner Claimed Unclaimed Matched Preempting  Drain Backfill
BkIdle

  X86_64/LINUX           3     0       0         3       0          0      0        0
     0

         Total           3     0       0         3       0          0      0        0
     0
```

Helped Aaron get onto our containers and Added him as an additional executor!

Goal: Parallel Jobs

- Wanted to try to run parallel jobs (even though this isn't exactly what HTCondor is for)
- Have to chose one submitter node to be the "dedicated scheduler" which handles submission of parallel jobs
  - All parallel jobs must be submitted from here. We chose Trevor's node.
- We need to "flip some switches" to get this to work…
- **Two of our "nodes" have already left, so we were unable to fully test this, so take everything we say with a grain of salt!**

Pointing to the Dedicated Scheduler

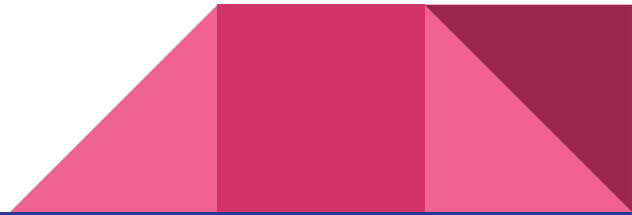
- We must denote the dedicated scheduler on by placing the following in the submitter config

```
Scheduler = "DedicatedScheduler@twnelson@cmswn3012.fnal.gov"
```

- We must point all of the executor nodes to the dedicated scheduler with this config option

```
DedicatedScheduler = "DedicatedScheduler@twnelson@cmswn3012.fnal.gov"

STARTD_ATTRS = $(STARTD_ATTRS), DedicatedScheduler
```



Setting Policy

- The executor nodes must have a policy for running the parallel jobs. We add the following to the executor configs

```
START     = Scheduler =?= $(DedicatedScheduler)

SUSPEND   = False

CONTINUE  = True

PREEMPT   = False

KILL      = False

WANT_SUSPEND   = False

WANT_VACATE    = False

RANK      = Scheduler =?= $(DedicatedScheduler)
```

Submitting A Parallel Job

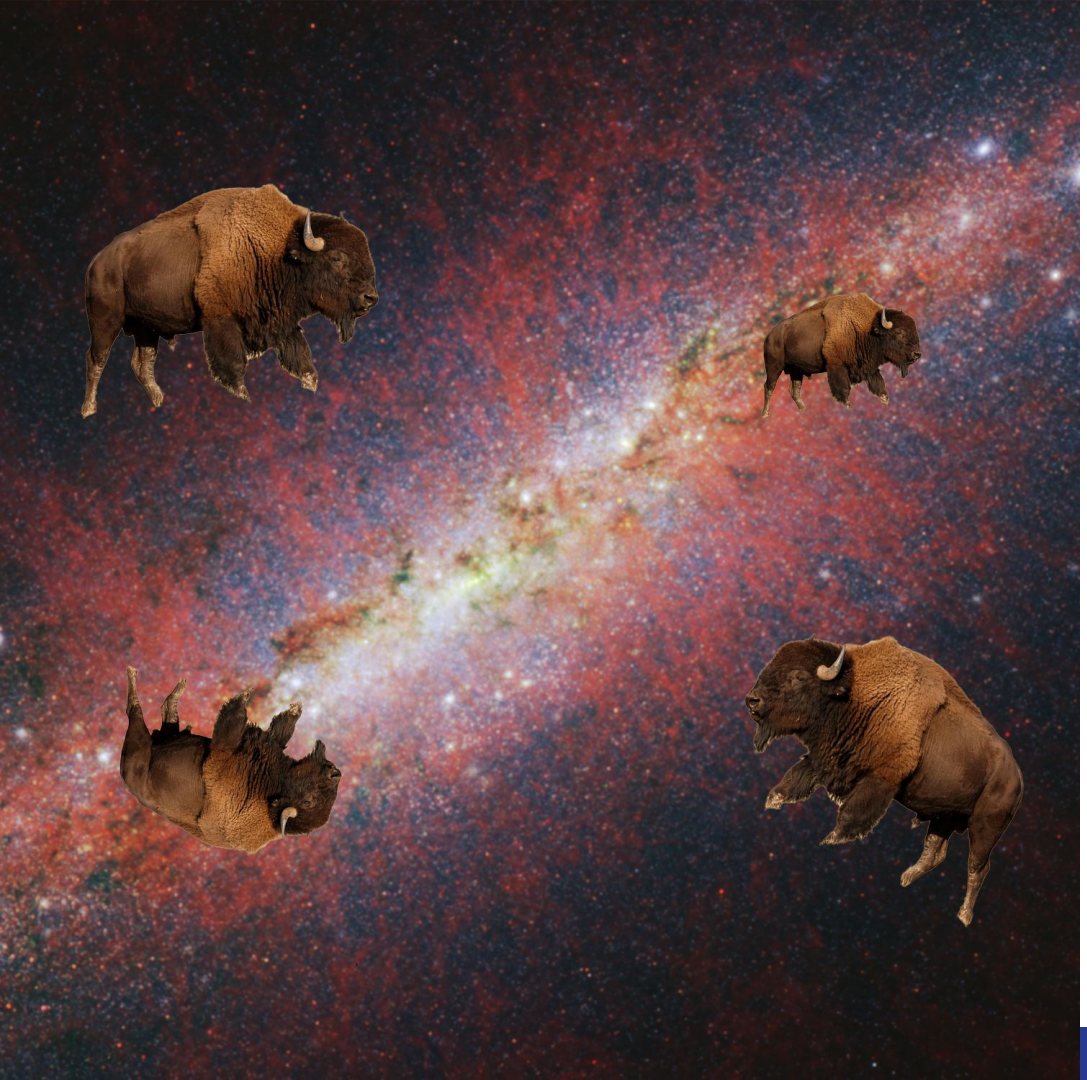
- On the user side, the submission must use the universe “parallel” and have a machine count
- The startd starts the executable on each machine at the same time, just like “mpirun”
- Each node has their own “Node” environment variable, just like MPI_Rank
- Can send the output from each node into different output files as well, or even use different input

```
########################################
## submit description file for a parallel program
## showing the $(Node) macro
########################################
universe = parallel
executable = /bin/cat
log = logfile
input = infile.$(Node)
output = outfile.$(Node)
error = errfile.$(Node)
machine_count = 4
should_transfer_files = IF_NEEDED
when_to_transfer_output = ON_EXIT
queue
```

Takeaways

- **We based this off of the documentation and weren't able to fully test this, so we may be missing steps**
- HTCondor isn't the best at parallel jobs because of queue prioritization and stuff
- Slurm, PBS, etc. are much easier to set up for parallel jobs and the queue prioritization is much more optimized for parallel jobs as well

JWST Instagram Post



EXTRAORDINARY! EXTRATERRESTRIAL BISON DISCOVERED BY THE JWST IN A DISTANT GALAXY - IS THIS FERMILAB'S FAULT??