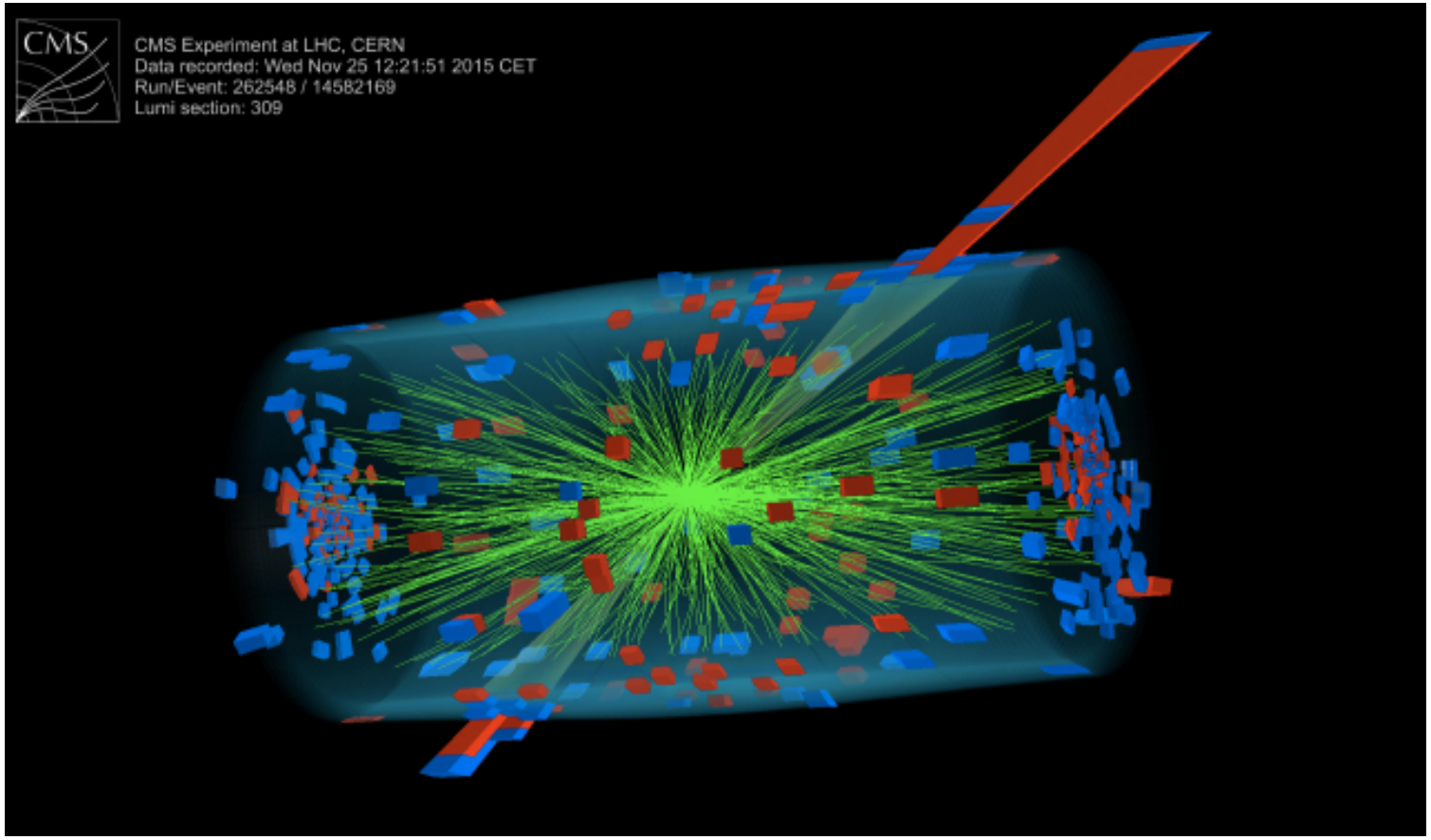
ACCRE
Advanced Computing Center for Research & Education

Computing at ACCRE

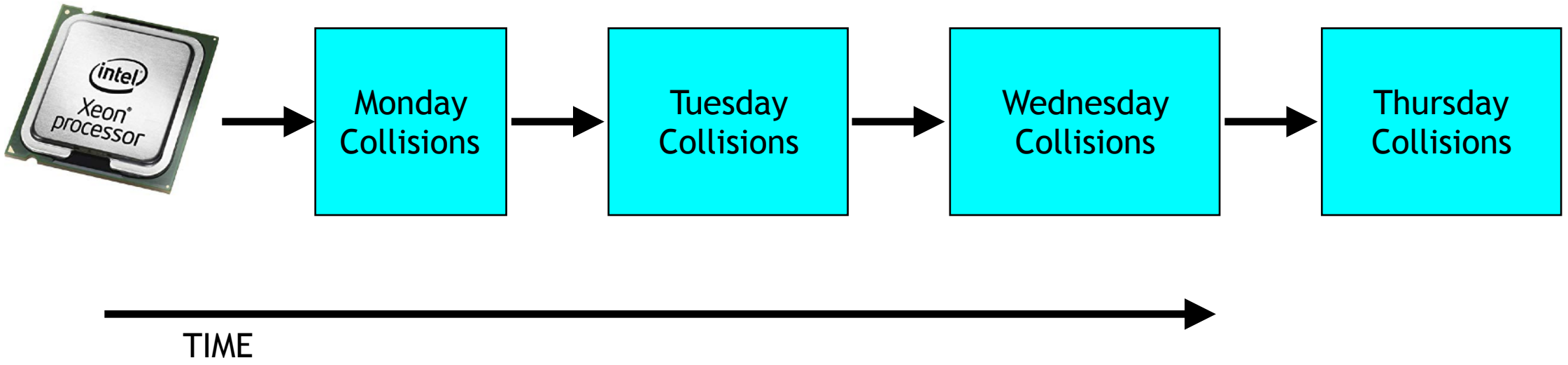
Parallel Computing Overview

Example: LHC Collision Processing



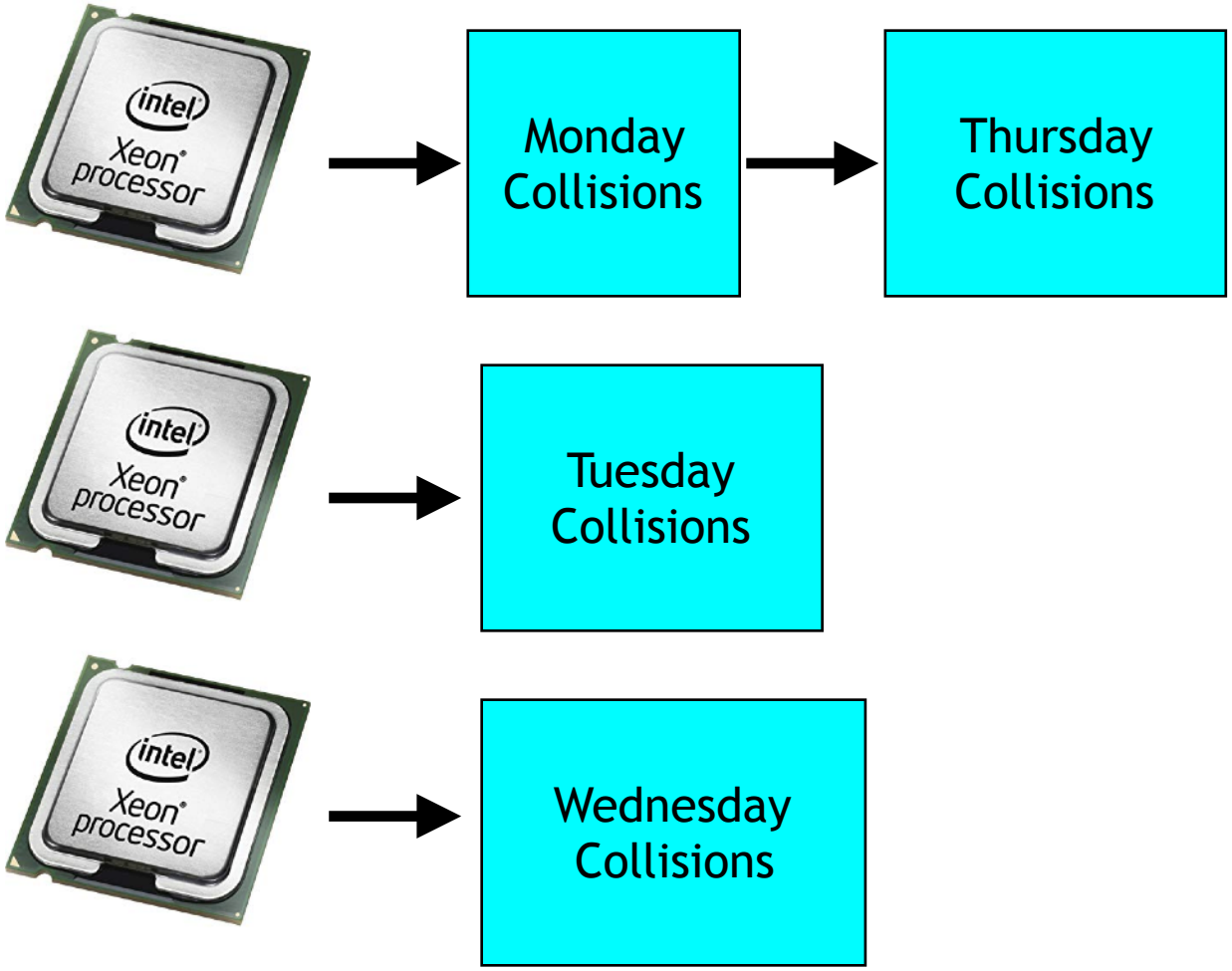
- Particle Accelerator produces millions of particle collision events each day
- Detector data for each event must be processed by computer to reconstruct trajectories
- Very CPU intensive process
- Each event is separate
- High Throughput Computing

Sequential Processing



- Slow to process each day of events one after another

"Embarrassingly" Parallel Processing

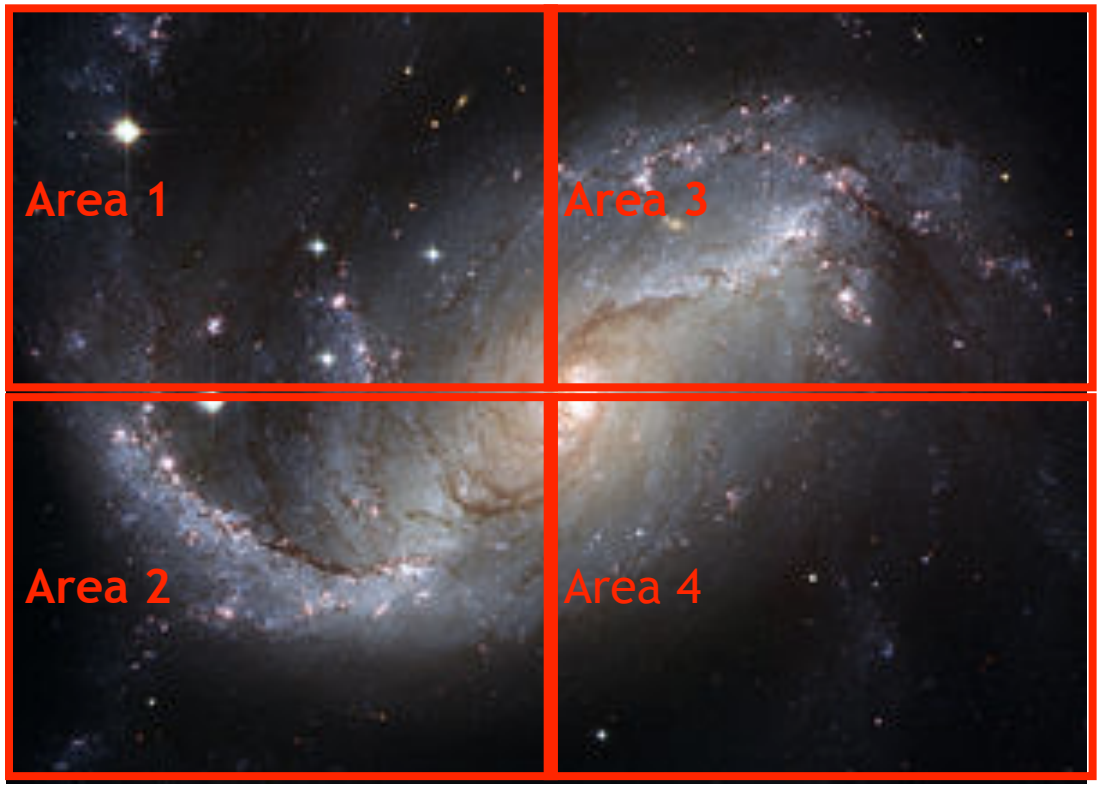


- Faster to process multiple days in parallel
- Any number of CPUs is helpful
- Tasks can start and stop at different times
- No communication between tasks, results for each task written to disk

Example: Galactic Simulation

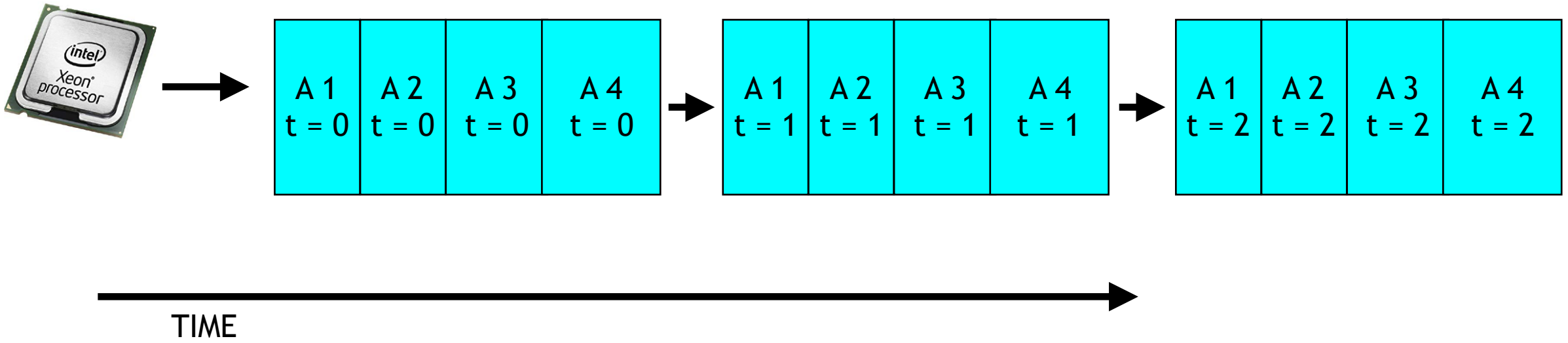
- Want to understand galactic formation
- Run detailed calculation of gravitational dynamics
- Very CPU intensive process
- Lots of individual stars or objects to simulate
- Each component may affect every other component
- High Performance Computing

Spatial Decomposition



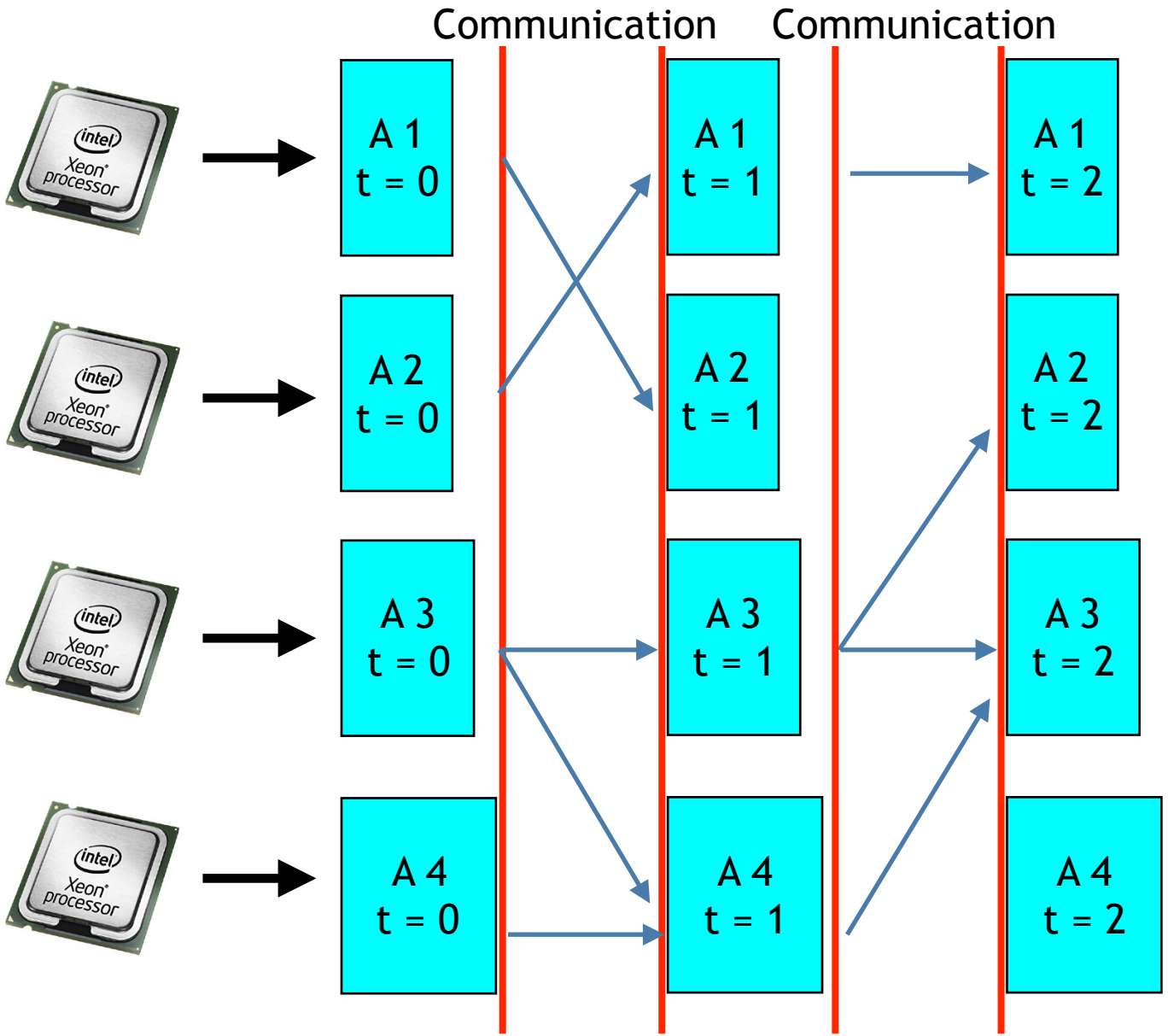
- Cut space into multiple areas
- Determine gravitational effect of each object within each area
- Approximate effects between areas

Sequential Processing



- Must simulate all 4 areas for each time step
- Move objects between areas after each step

Parallel Message Passing



- All areas must complete before next iteration
- Time required to communicate after each iteration
- Less CPUs than areas may lead to delays of entire simulation
- CPUs may be forced to spend time waiting for the others

ACCRE OVERVIEW

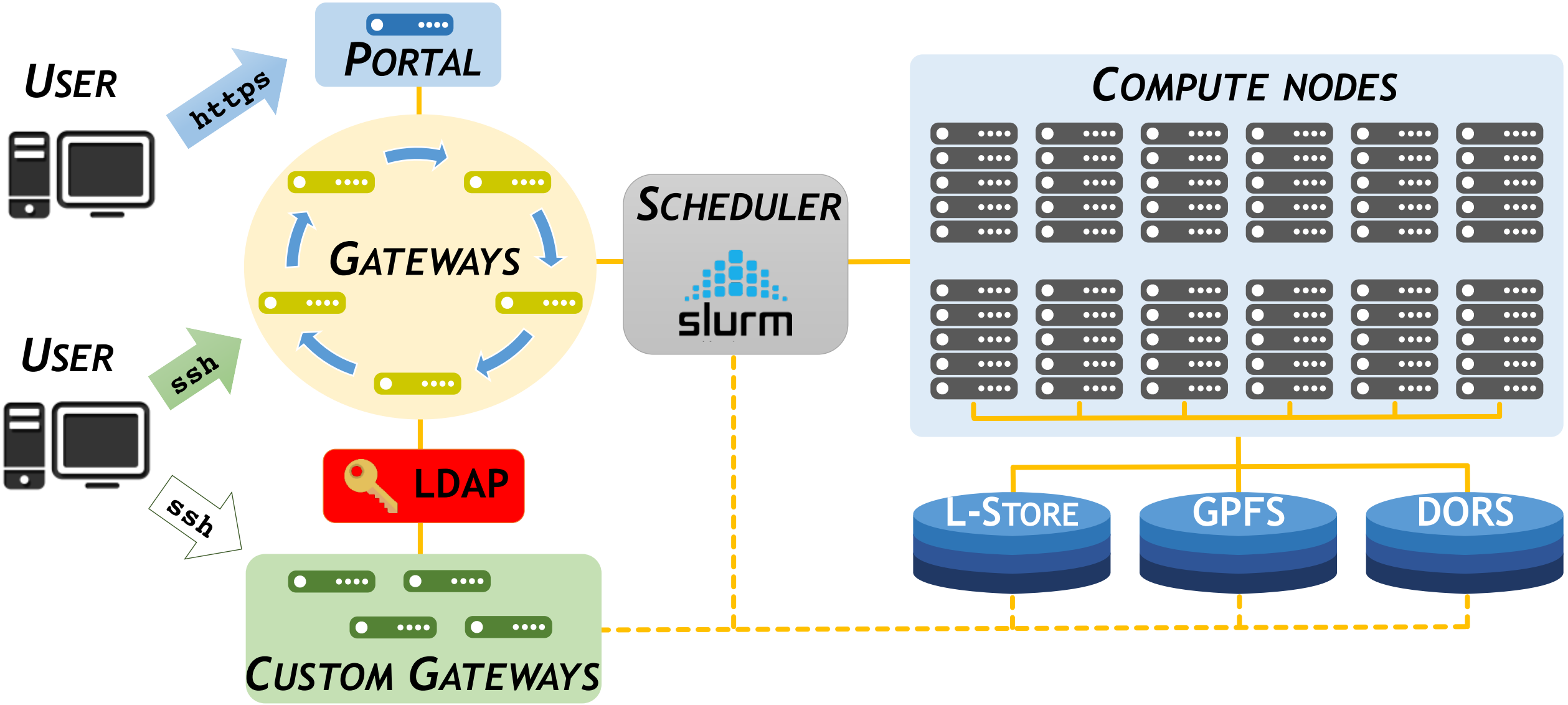
ACCRE Overview

- ACCRE - Advanced Computing Center For Research and Education
- Centralized computing infrastructure for Vanderbilt researchers
- Operates as a co-op in which researchers share hardware
- ~10k CPU cores
- ~200 GPUs
- ~10PB disk storage + tape backups
- Optimized Scientific Software Stack
- Batch Job Scheduler
- Interactive resources (Jupyter, etc.)
- Staff of ~10

Using ACCRE vs Using Your Own Hardware

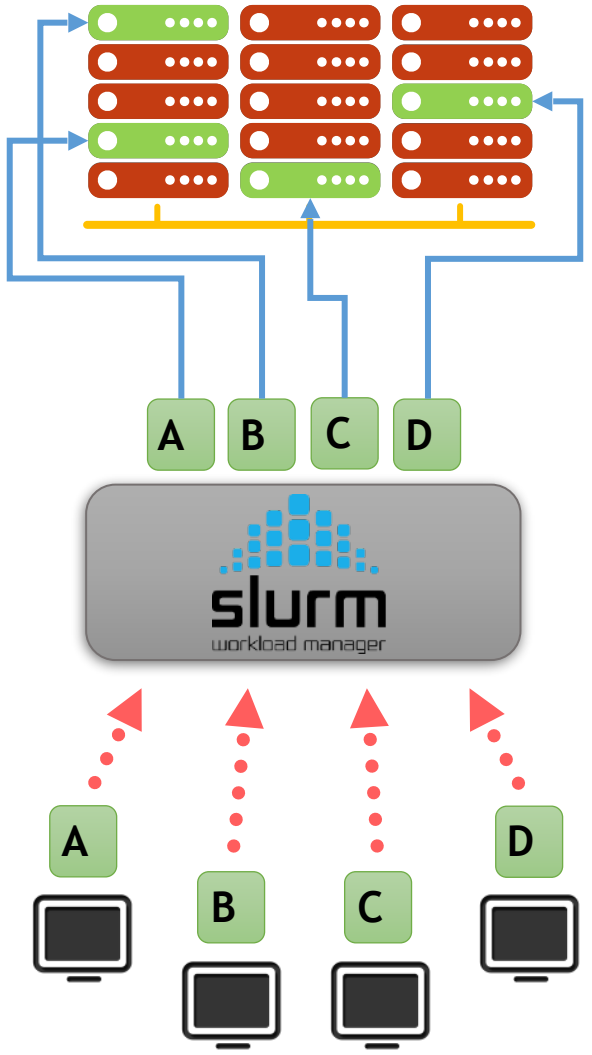
- Using your own hardware:
  - can use all resources immediately
  - have to set up software, system, and networking yourself
  - full administrative access (root)
- Using ACCRE:
  - must schedule resource requirements
  - can "burst" to use more resources than you own
  - dedicated staff maintain system and software stack
  - no administrative access (regular user)

ACCRE ARCHITECTURE



THE SCHEDULER

1. Execute user's workloads in the right priority order
2. Provide requested resources on compute nodes
3. Optimize cluster utilization



ACCRE is a Heterogeneous Cluster

- Different Memory Configurations and CPU Core Counts
  - Nodes with 64GB, 128GB, 192GB, 256GB, and 384GB
  - Between 8 and 32 CPU-cores per node
- Different Intel CPU Architecture Families
  - Variable clock speed, L1/2/3 Cache Memory
  - Additional Instruction Sets on Newer CPUs
- Specialized Accelerated Nodes
  - Nvidia 4x GPU Nodes (Maxwell, Pascal, Turing)

ACCRE is a Heterogeneous Cluster

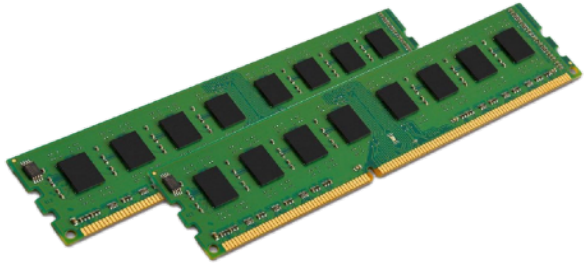
- Heterogeneity is ok for high-throughput computing (HTC) tasks
  - Don't care if jobs start at the same time
  - Don't care if some jobs are slower than others
- Not so great for distributed simulations i.e. high-performance computing (HPC) tasks
  - Need to synchronize between iterations
  - Limited to speed of slowest nodes
- ACCRE cluster users can choose to run on all architectures or specific architectures

ACCRE Cluster compute nodes

Regular nodes

Dual multicore CPUs

Random Access Memory



Newer

Older

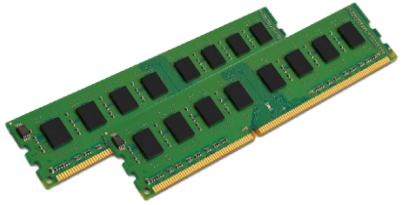
Family	No. of cores	RAM / GB	No. of nodes
Skylake	16	256	41
	24	128	52
Haswell	12	128	41
	16	128	120
		256	50
Sandy Bridge	12	64	31
		96	2
		128	193
		256	4
	16	128	3
Westmere	8	128	22
	12	48	16
Total	8,292	82,432	575

THE COMPUTE NODES

Accelerated nodes

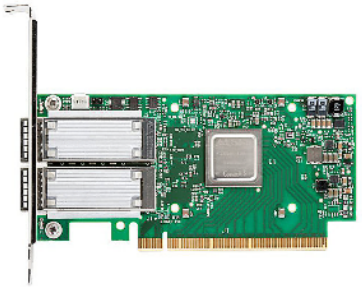
Dual multicore CPUs

Random Access Memory



4 x Nvidia GPU

25/40 Gbit/s RoCE Network



Newer

Older

Family	*No. of cores*	*RAM / GB*	*No. of nodes (GPUs)*
Nvidia Turing ***Intel Skylake***	**24**	*384*	21 (84)
Nvidia Pascal ***Intel Broadwell***	**8**	*256*	24 (96)
Nvidia Maxwell ***Intel Haswell***	**12**	*128*	10 (40)
Total	**816**	**15,488**	**55 (220)**

ACCRE
Advanced Computing Center for Research & Education

Submitting Batch Jobs

Determining Requirements

- # of tasks
- # of cpu cores per task
- Memory (GB) per node or core
- Time allowed to complete job

Optimizing requirements results in jobs being scheduled sooner

Optimizing requirements makes better use of the cluster

Example Batch Job Script

A **batch job** consists of a sequence of commands listed in a file with the purpose of being interpreted as a single program.

SHEBANG

- Specify the script interpreter (Bash)
- Must be the first line!

SLURM DIRECTIVES

- Start with "#SBATCH":
  Parsed by Slurm but ignored by Bash.
- Can be separated by spaces.
- Comments between and after directives are allowed.
- Must be before actual commands!

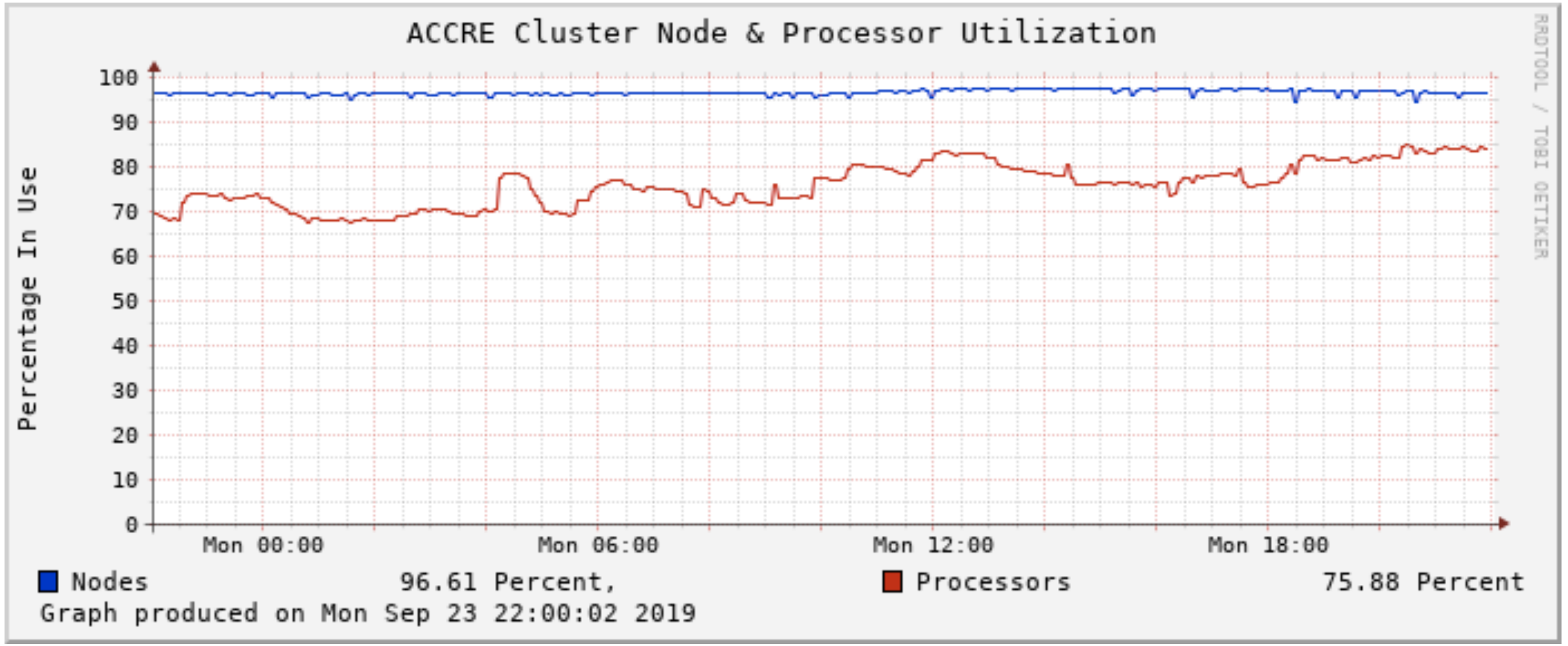
SCRIPT COMMANDS

- Commands you want to execute on the compute nodes.

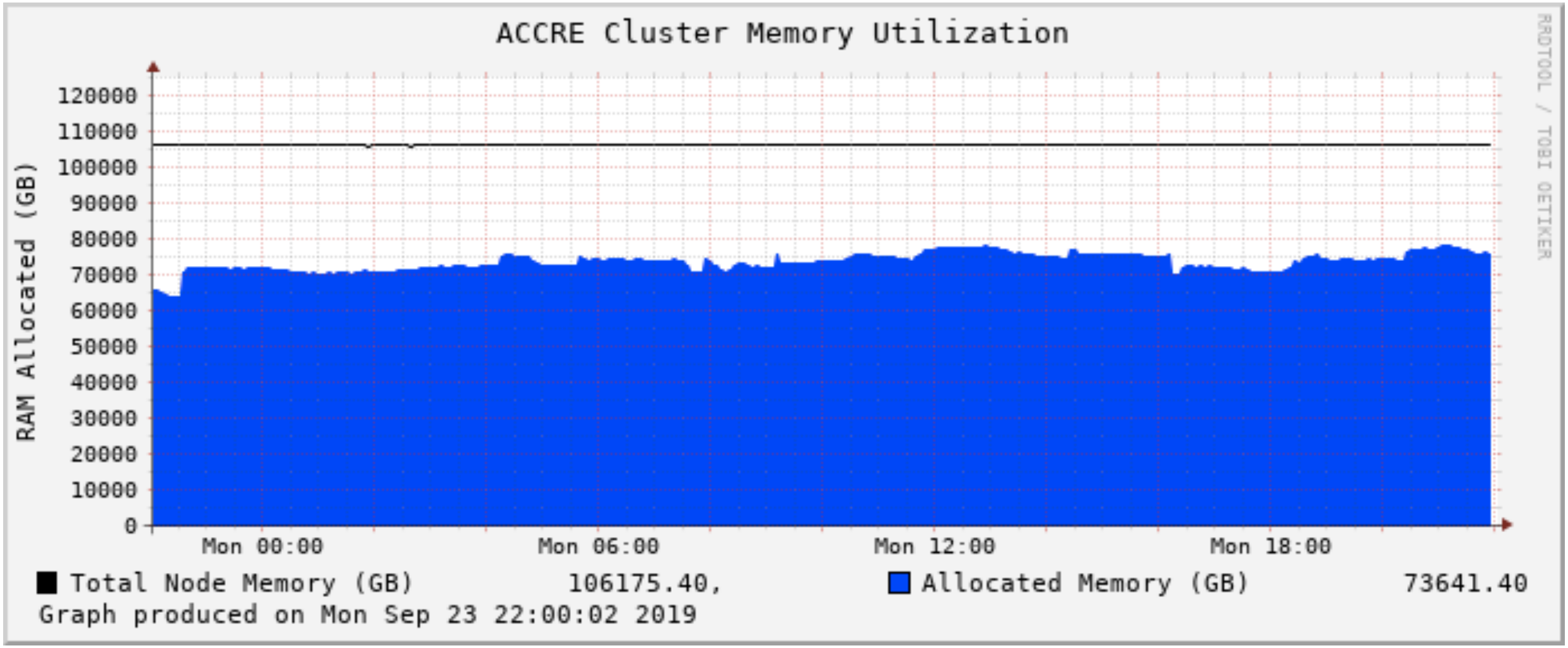
myjob.slurm

```
#!/bin/bash

#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --mem=1G
#SBATCH --time=1-06:30:00
#SBATCH --job-name=myjob
#SBATCH --output=myjob.out


# Just a comment
module load GCC Python
python myscript.py
```

Cluster Usage Plots



Cluster Usage Plots



Cluster Usage Plots

- Plots show CPU cores and memory requested and allocated to jobs.
- They do not show what the jobs are actually using
- They do not show how many jobs are completing successfully
- Better optimization means less hardware purchases required
- More research for less $$$

Demo Time

ACCRE
Advanced Computing Center for Research & Education

- Demo Cluster Login
- Demo Running an Example Job
- Demo Running a Job Array
- Multiple cores vs multiple nodes

Analyzing Job Usage

- Want jobs to complete successfully
- Want jobs to use high percentage of memory requested
- Want CMS jobs to last > 30 minutes and avoid internal failure, minimize time lost in scheduling resources

Job Record Format

```
JOBID,ACCOUNT,USER,REQMEM,USEDMEM,REQTIME,USEDTIME,NODES,CPUS,PARTITION,EXITCODE,STATE,NODELIST
7148570_177,sommerfeld,treena,16384Mn,1892.96M,5-00:00:00,2-12:09:36,1,1,production,0:0,COMPLETED,cn1520
```

- Job records are given as a CSV file with one row for each job

Job Record Format

```
JOBID,ACCOUNT,USER,REQMEM,USEDMEM,REQTIME,USEDTIME,NODES,CPUS,PARTITION,EXITCODE,STATE,NODELIST
7148570_177,sommerfeld,treena,16384Mn,1892.96M,5-00:00:00,2-12:09:36,1,1,production,0:0,COMPLETED,cn1520
```

- Each job gets a unique ID
- For a large group of similar jobs, a job-array may be used
- Array tasks have numbers after an underscore
- Array tasks treated as individual jobs

Job Record Format

```
JOBID,ACCOUNT,USER,REQMEM,USEDMEM,REQTIME,USEDTIME,NODES,CPUS,PARTITION,EXITCODE,STATE,NODELIST
7148570_177,sommerfeld,treena,16384Mn,1892.96M,5-00:00:00,2-12:09:36,1,1,production,0:0,COMPLETED,cn1520
```

- Each research group has one or more accounts
- ACCRE accounts are anonymized in this dataset
- Job fairshare and priority is counted by account

Job Record Format

```
JOBID,ACCOUNT,USER,REQMEM,USEDMEM,REQTIME,USEDTIME,NODES,CPUS,PARTITION,EXITCODE,STATE,NODELIST
7148570_177,sommerfeld,treena,16384Mn,1892.96M,5-00:00:00,2-12:09:36,1,1,production,0:0,COMPLETED,cn1520
```

- Users are individual researchers
- ACCRE users are anonymized in this dataset

Job Record Format

```
JOBID,ACCOUNT,USER,REQMEM,USEDMEM,REQTIME,USEDTIME,NODES,CPUS,PARTITION,EXITCODE,STATE,NODELIST
7148570_177,sommerfeld,treena,16384Mn,1892.96M,5-00:00:00,2-12:09:36,1,1,production,0:0,COMPLETED,cn1520
```

- Requested Memory is in Megabytes
- May be per-core (Mc) or per-node (Mn)
- Need to convert to per-core for consistency

Job Record Format

```
JOBID,ACCOUNT,USER,REQMEM,USEDMEM,REQTIME,USEDTIME,NODES,CPUS,PARTITION,EXITCODE,STATE,NODELIST
7148570_177,sommerfeld,treena,16384Mn,1892.96M,5-00:00:00,2-12:09:36,1,1,production,0:0,COMPLETED,cn1520
```

- Used Memory is in MB, per-node
- Trailing "M" may be missing if memory usage is zero
- Need to convert to per-core for consistency
- Low % of requested memory is an inefficient use of resources!

Job Record Format

```
JOBID,ACCOUNT,USER,REQMEM,USEDMEM,REQTIME,USEDTIME,NODES,CPUS,PARTITION,EXITCODE,STATE,NODELIST
7148570_177,sommerfeld,treena,16384Mn,1892.96M,5-00:00:00,2-12:09:36,1,1,production,0:0,COMPLETED,cn1520
```

- Requested time is in d-hh:mm:ss or just hh:mm:ss

Job Record Format

```
JOBID,ACCOUNT,USER,REQMEM,USEDMEM,REQTIME,USEDTIME,NODES,CPUS,PARTITION,EXITCODE,STATE,NODELIST
7148570_177,sommerfeld,treena,16384Mn,1892.96M,5-00:00:00,2-12:09:36,1,1,production,0:0,COMPLETED,cn1520
```

- Used time is in d-hh:mm:ss or just hh:mm:ss
- Watch out for really short jobs! Bad use of cluster resources, and in the case of the CMS account, an indication of internal failure

Job Record Format

```
JOBID,ACCOUNT,USER,REQMEM,USEDMEM,REQTIME,USEDTIME,NODES,CPUS,PARTITION,EXITCODE,STATE,NODELIST
7148570_177,sommerfeld,treena,16384Mn,1892.96M,5-00:00:00,2-12:09:36,1,1,production,0:0,COMPLETED,cn1520
```

- Nodes is number of servers used for this job
- Multi-node jobs are uncommon at ACCRE
- For multi-node jobs memory usage is the maximum over all nodes

Job Record Format

```
JOBID,ACCOUNT,USER,REQMEM,USEDMEM,REQTIME,USEDTIME,NODES,CPUS,PARTITION,EXITCODE,STATE,NODELIST
7148570_177,sommerfeld,treena,16384Mn,1892.96M,5-00:00:00,2-12:09:36,1,1,production,0:0,COMPLETED,cn1520
```

- CPUs is the total number of CPU-cores allocated to the job
- For multi-node jobs this includes all nodes
-

Job Record Format

```
JOBID,ACCOUNT,USER,REQMEM,USEDMEM,REQTIME,USEDTIME,NODES,CPUS,PARTITION,EXITCODE,STATE,NODELIST
7148570_177,sommerfeld,treena,16384Mn,1892.96M,5-00:00:00,2-12:09:36,1,1,production,0:0,COMPLETED,cn1520
```

- Most jobs run in the production partition
- Ignore jobs in the "debug" partition, these are test jobs and expected to be short and use a low % of requested memory
- The "maxwell", "pascal", and "turing" partitions are for GPU resources, these should be analyzed separately or ignored
- The “nogpfs” partition is of interest only to CMS

Job Record Format

```
JOBID,ACCOUNT,USER,REQMEM,USEDMEM,REQTIME,USEDTIME,NODES,CPUS,PARTITION,EXITCODE,STATE,NODELIST
7148570_177,sommerfeld,treena,16384Mn,1892.96M,5-00:00:00,2-12:09:36,1,1,production,0:0,COMPLETED,cn1520
```

- Exit code should be "0:0" for successful jobs
- Lots of long-running failed jobs are a waste of resources
- Can ignore failed jobs when analyzing average memory usage
- Nonzero on the left of the “:” means an error in the users software, non-zero on the right means an issue with the job itself or node (i.e. out of memory, node failure, cancelled, etc)

Job Record Format

```
JOBID,ACCOUNT,USER,REQMEM,USEDMEM,REQTIME,USEDTIME,NODES,CPUS,PARTITION,EXITCODE,STATE,NODELIST
7148570_177,sommerfeld,treena,16384Mn,1892.96M,5-00:00:00,2-12:09:36,1,1,production,0:0,COMPLETED,cn1520
```

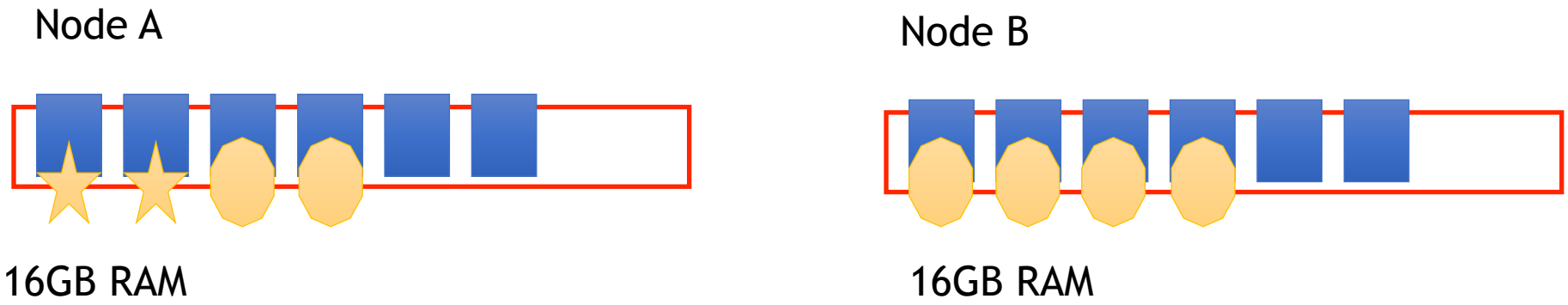
- State should be "COMPLTETED" for successful jobs
- A “CANCELLED” state indicates the user intentionally stopped the job
- A “FAILED” state indicates a problem running the job or overconsumption of resources

Job Record Format

```
JOBID,ACCOUNT,USER,REQMEM,USEDMEM,REQTIME,USEDTIME,NODES,CPUS,PARTITION,EXITCODE,STATE,NODELIST
7148570_177,sommerfeld,treena,16384Mn,1892.96M,5-00:00:00,2-12:09:36,1,1,production,0:0,COMPLETED,cn1520
```

- Nodelist is a comma separated list of nodes the job ran on. Sequentially numbered nodes may be listed with brackets

Job Record Format




```
JOBID,ACCOUNT,USER,REQMEM,USEDMEM,REQTIME,USEDTIME,NODES,CPUS,PARTITION,EXITCODE,…

7148570,sommerfeld,treena,16384Mn,1892.96M,5-00:00:00,2-12:09:36,2,8,production,0:0,…
```

I submit a job:

—nodes = 2
—tasks-per-node=4
—mem = 16Gn

Memory per core (requested) = 4096Mc Memory Used per core (approx) = 473Mc

Analysis Questions

Q: What groups are best optimizing their memory usage in terms of percent of actual memory used of the memory requested for a job? What is the average percent for each group?

Analysis Questions

Q: Optimizing memory is more important for longer running jobs then shorter running jobs as the resources are tied up for longer. If jobs are weighted by runtime, what is the average percent of memory used of the requested memory for each group?

Analysis Questions

Q: We are concerned with potentially unreliable nodes which are not being detected by our routine monitoring. Of the failed jobs, do any nodes show up unusually often?

- Ignore debug partition
- Look for failed jobs

Analysis Questions

Q: The CMS collaboration submits jobs that will run internal diagnostics and intentionally end early in 30 minutes, how often is this happening (what percent of CMS jobs), and does it happen on the same nodes repeatedly?

- Check both “production” and “nogpfs” partitions
- Look for commonly failing nodes, compare with other failed jobs