HPC Concepts

Queues Schedulers, and workflow design for shared systems.

Overview

- Types
  - Of Computing systems
  - Of Jobs
- Limitations
  - Compute
  - I/O
- Cluster components
- Seperation
- Queuing
  - Queue Partitioning
  - Queue Filling
- Workflow organization as solution
- Examples

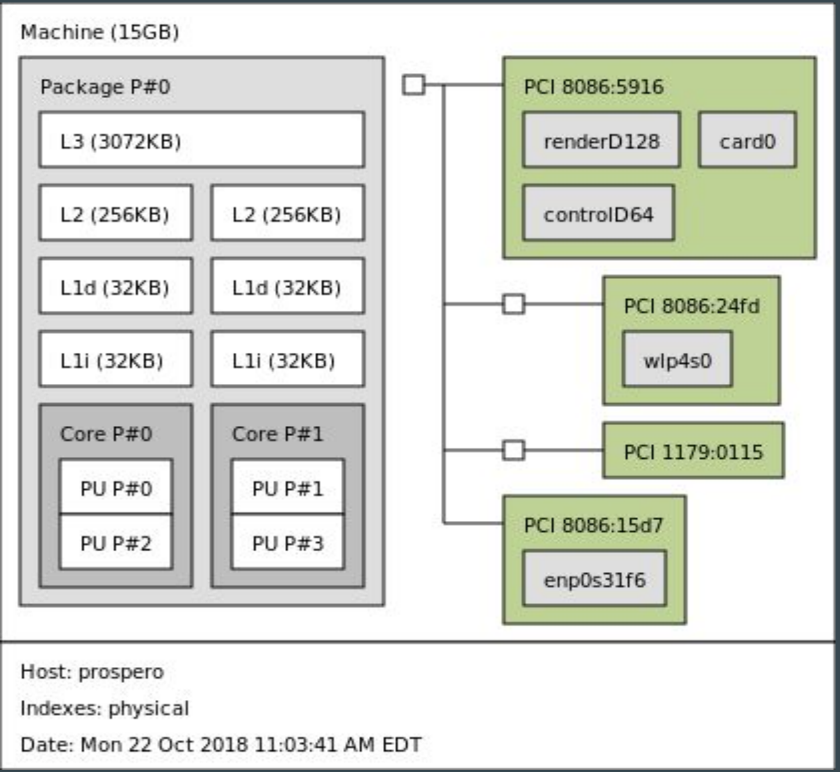
What is a HPC?

What is a HPC?

a device for turning compute-bound problems into I/O-bound problems.

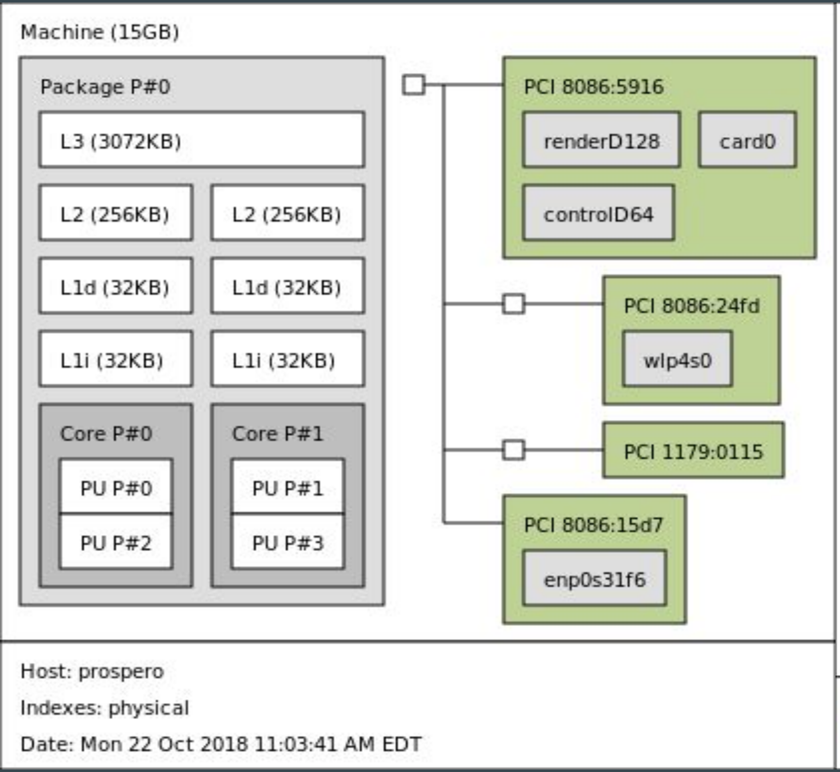
– Ken Batcher

More powerful CPU’s,   More Chips Per board

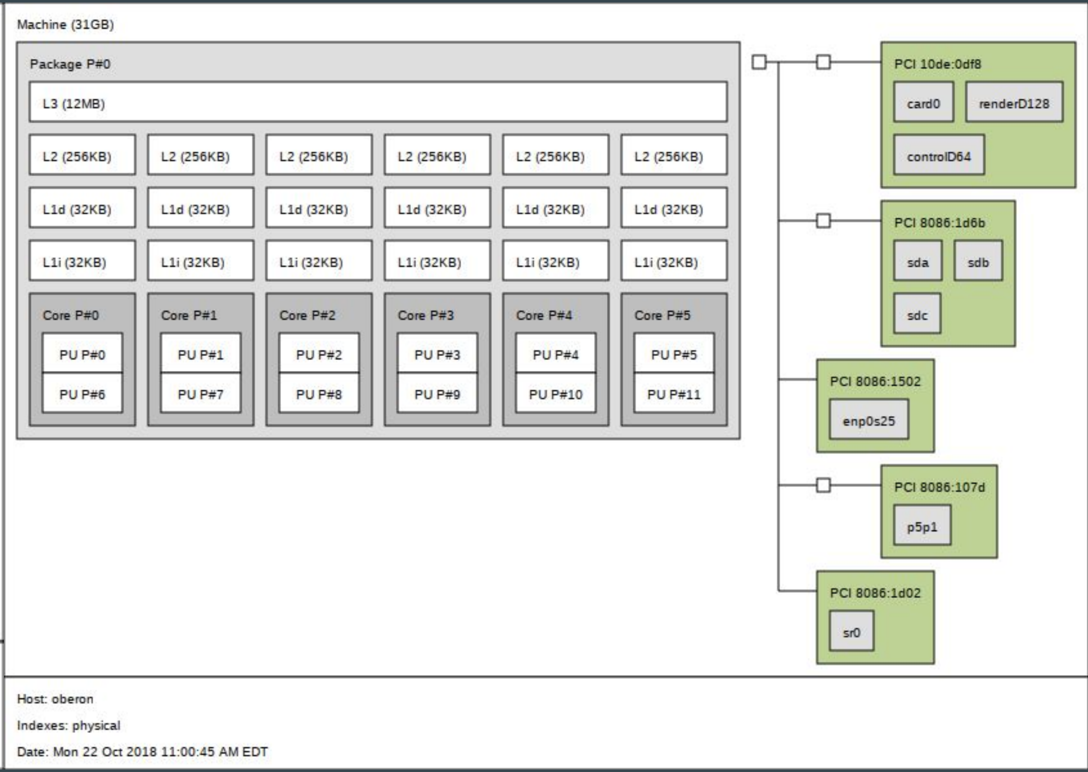


‘Business Laptop’

More powerful CPU’s, More Chips Per board

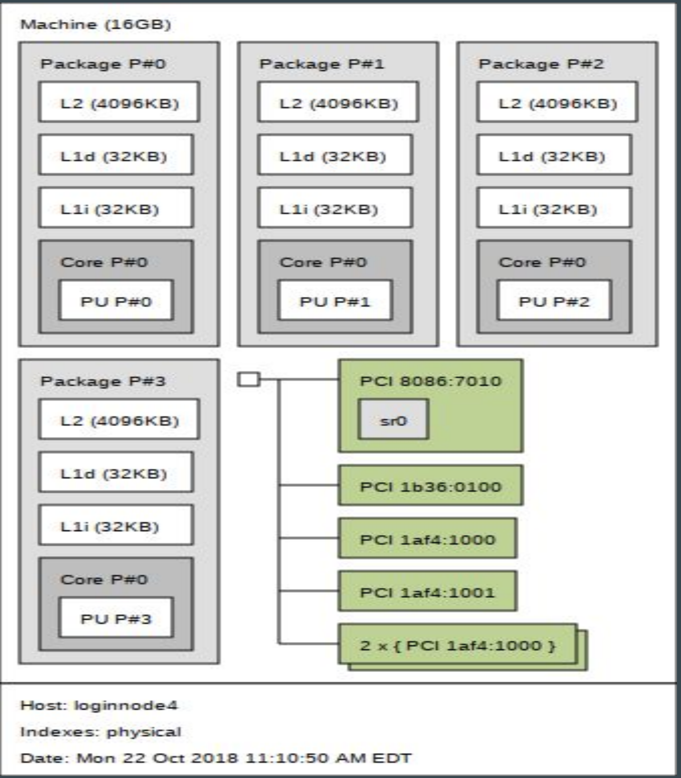


‘Business Laptop’

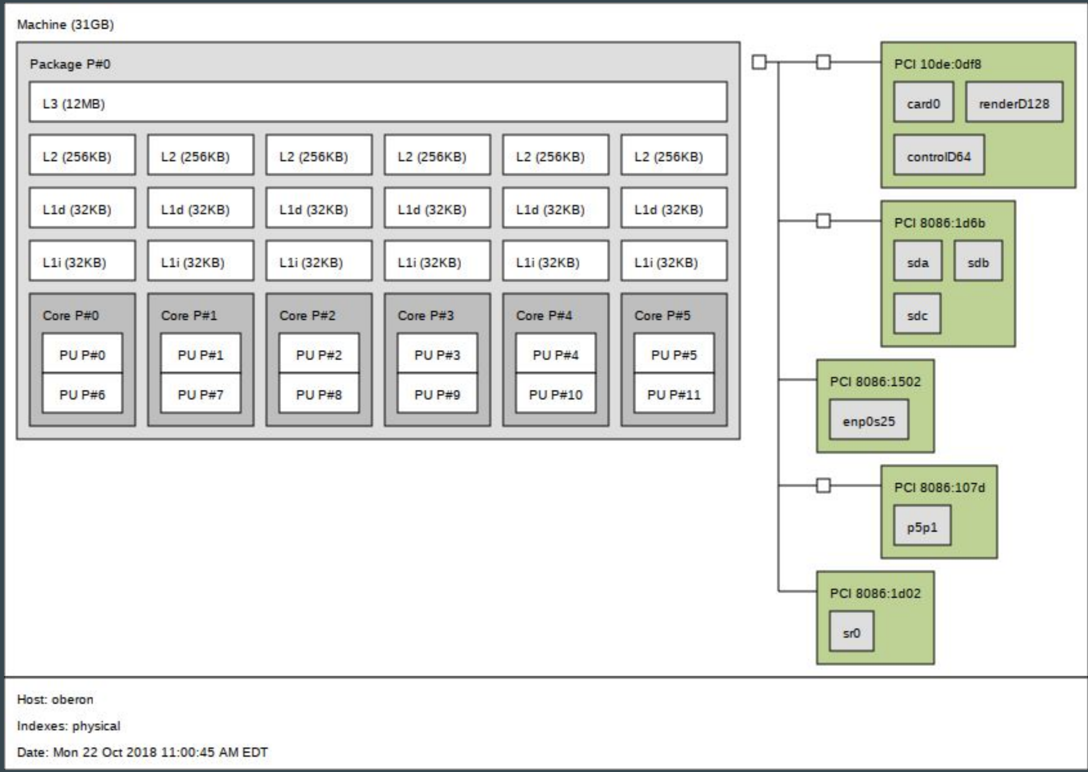


‘Mid-level Workstation’

More powerful CPU's, More Chips Per board - Reorganized

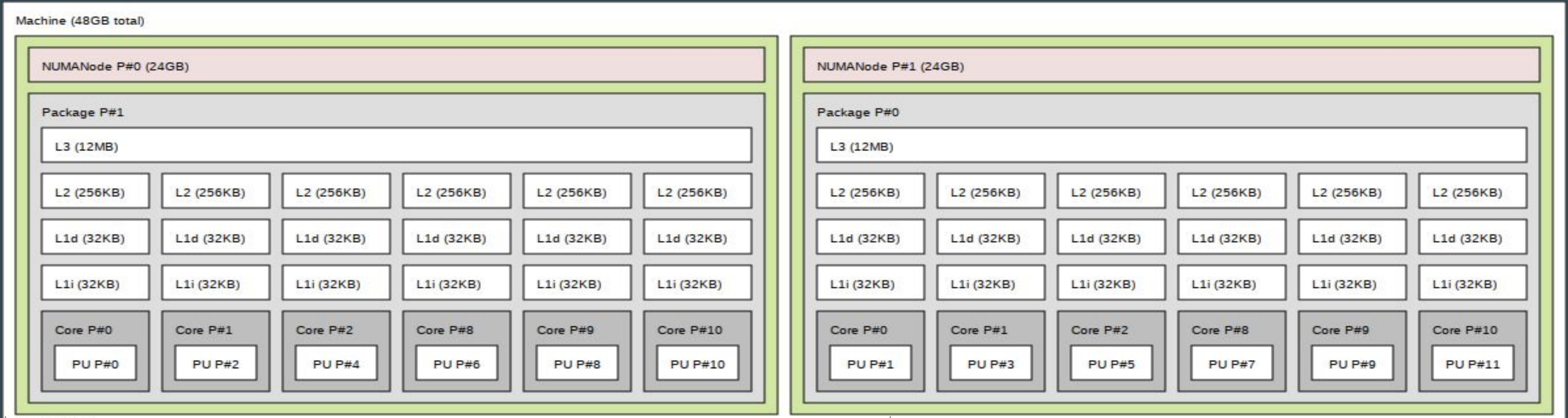


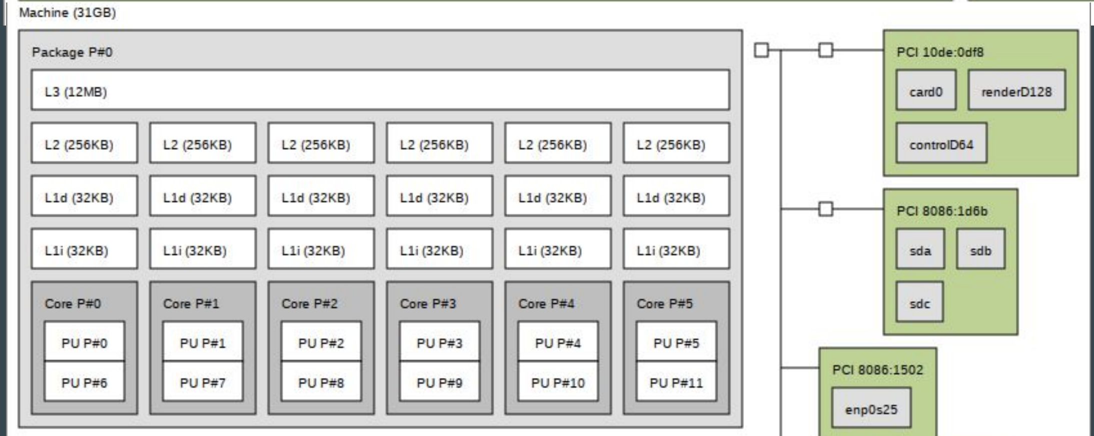
Head node



'Mid-level Workstation'

More powerful CPU’s,   More Chips Per board - Reorganized

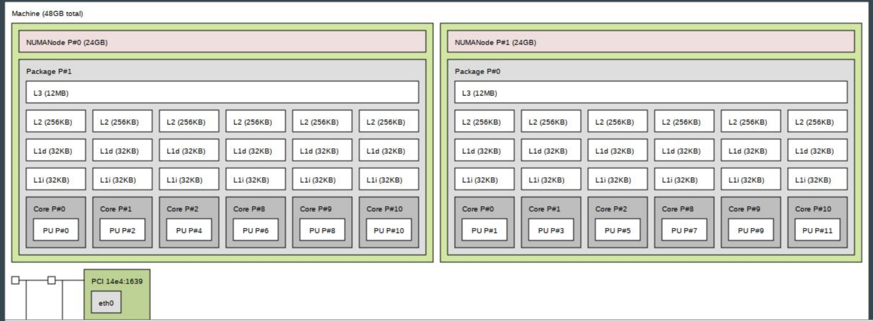




Small worker node

Same
‘Mid-level Workstation’

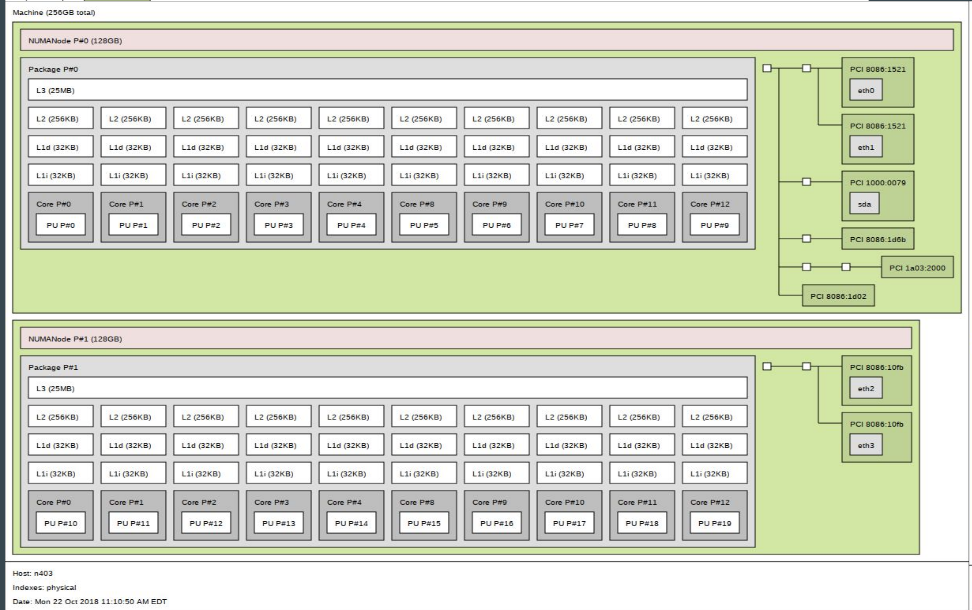
More powerful CPU's, More Chips Per board - Reorganized



Small worker node
2x 6 cores
2x 24Gb RAM

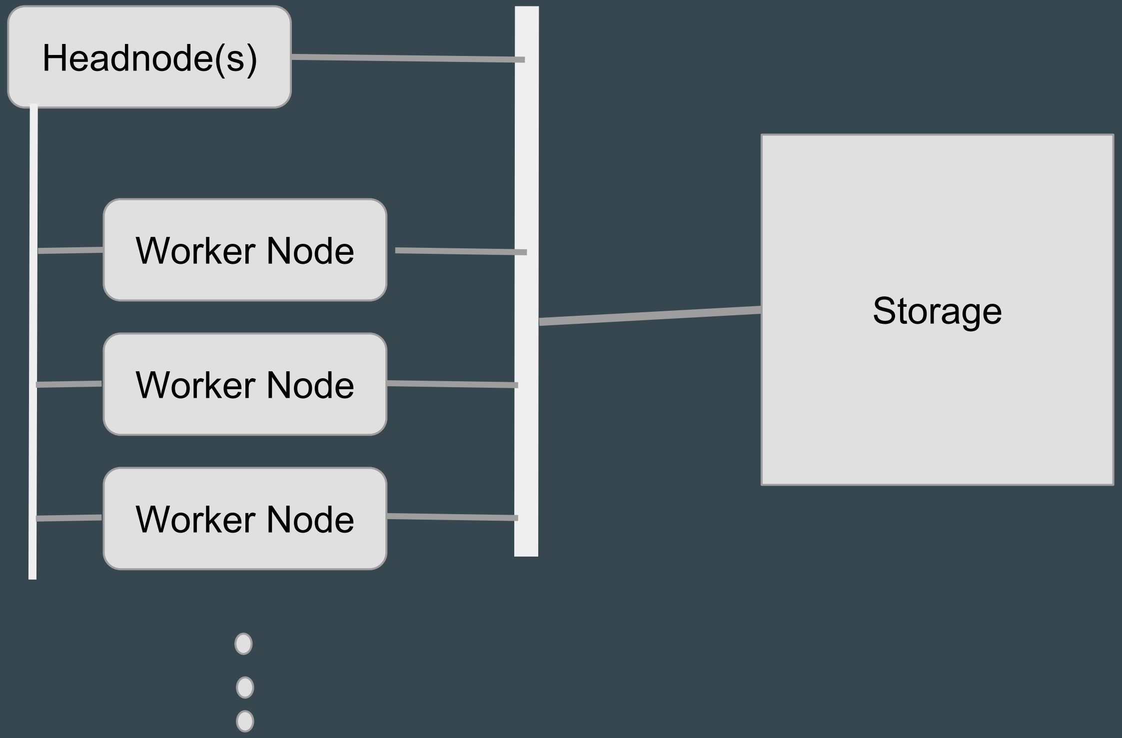
Mid-size worker node
2x 12 cores
2x 128 Gb RAM

Big-compute worker node
2x 14 cores
2x 128 Gb RAM



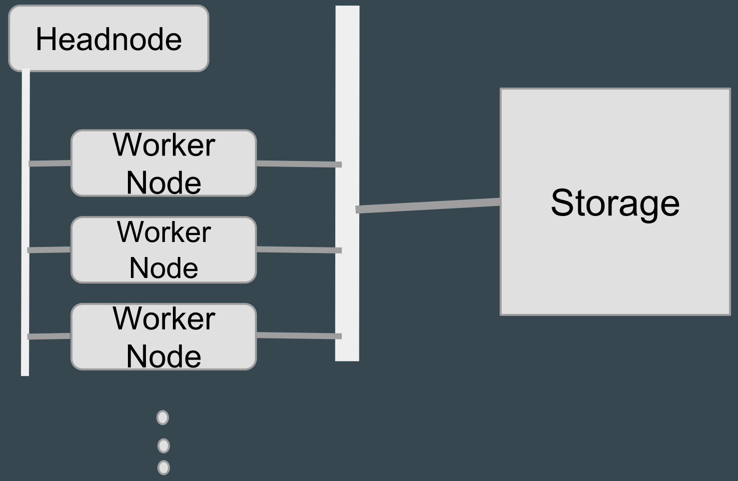
More Interconnects

Private LAN



Types of HPC

More Interconnects

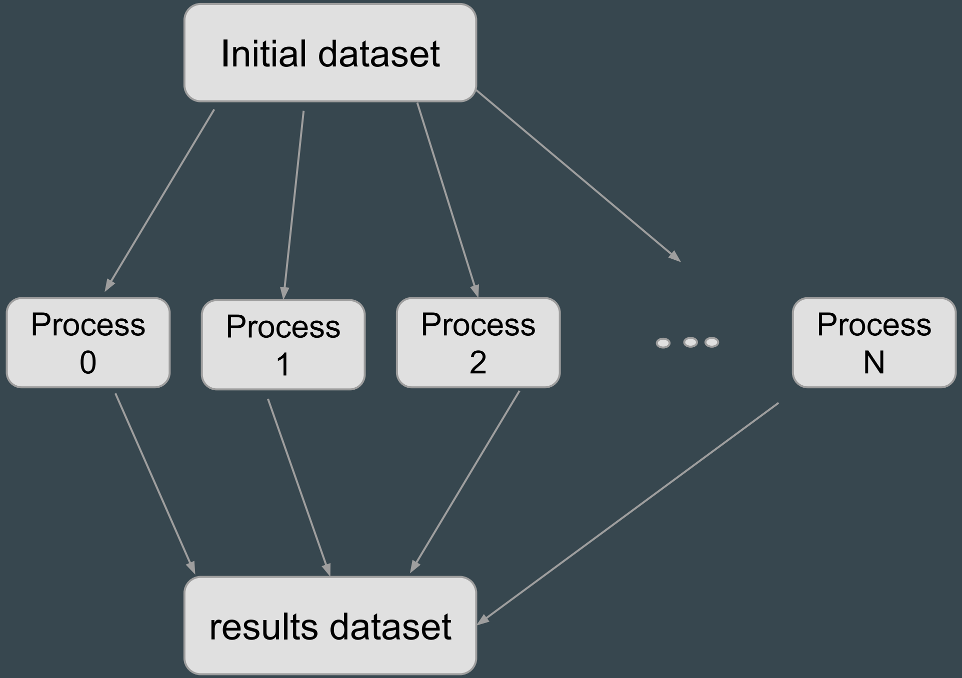


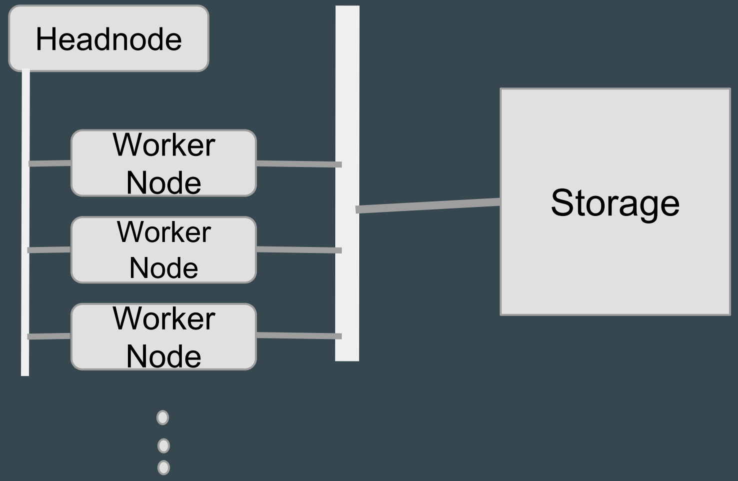
Types of HPC

More Interconnects

“Pleasingly Parallel”

Minimal complexity

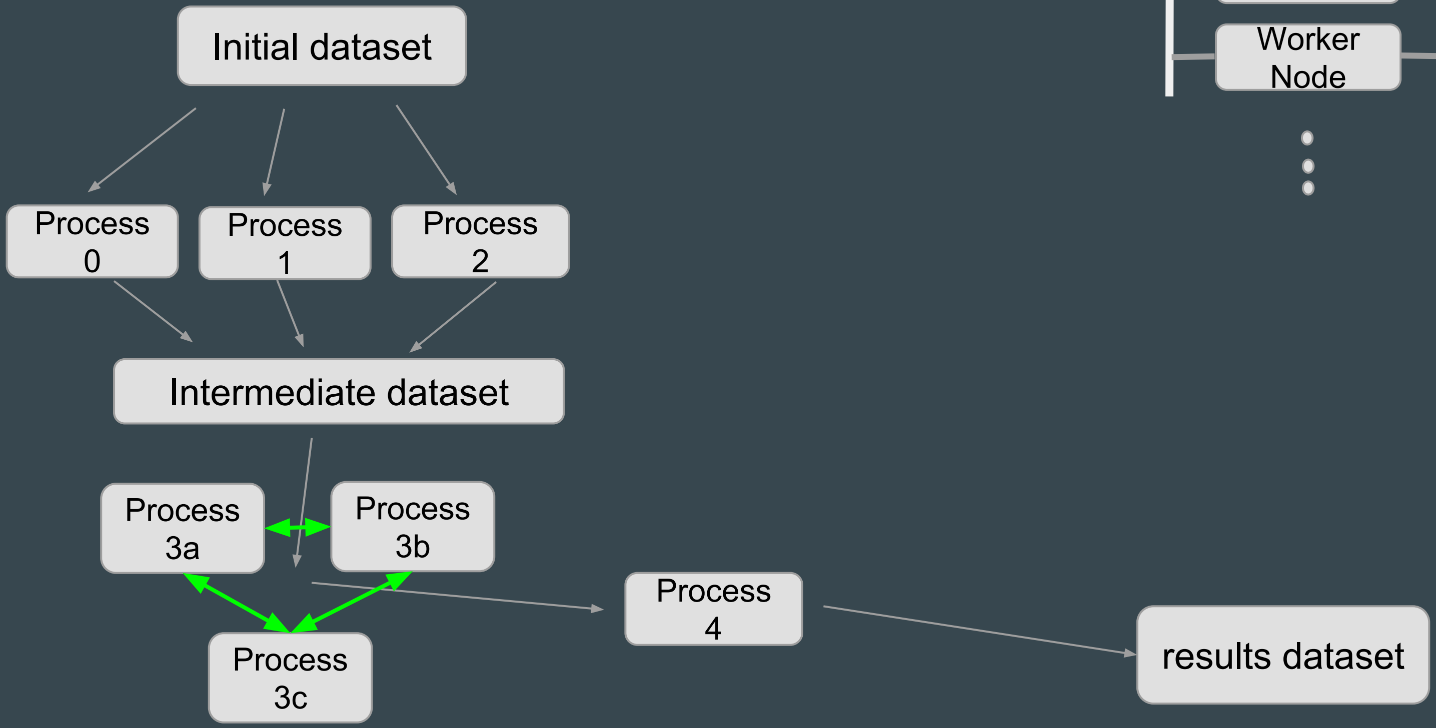


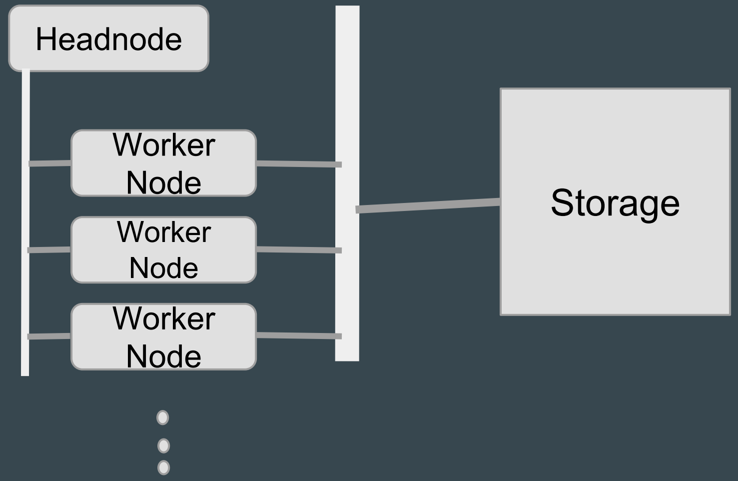


Types of HPC

More Interconnects

Loosely coupled workflows

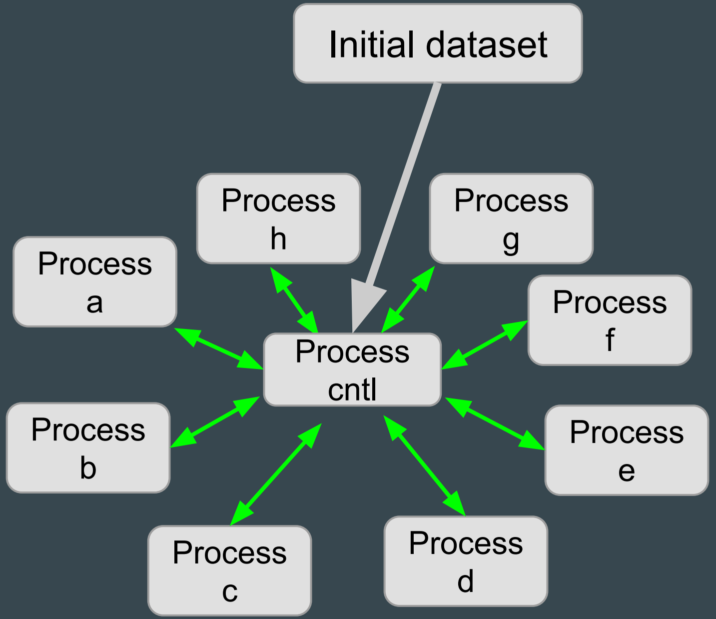


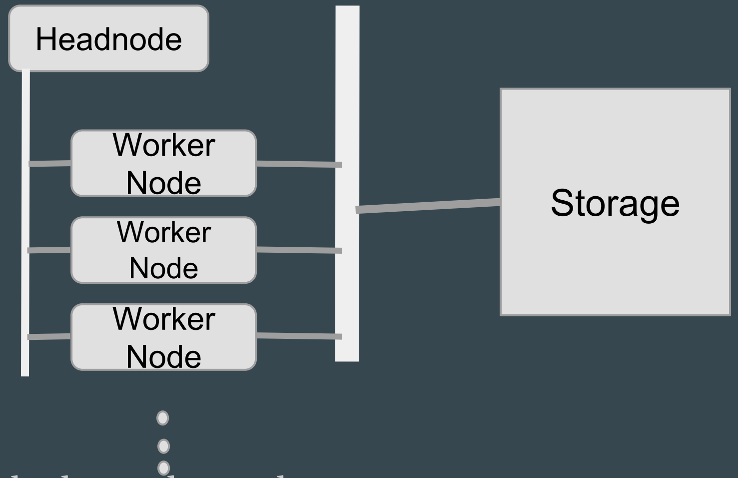


Types of HPC

More Interconnects

Loosely coupled workflows





Loosely coupled jobs have dependent processes, but distribute the data according to the task the are

Dependent in terms of task order and

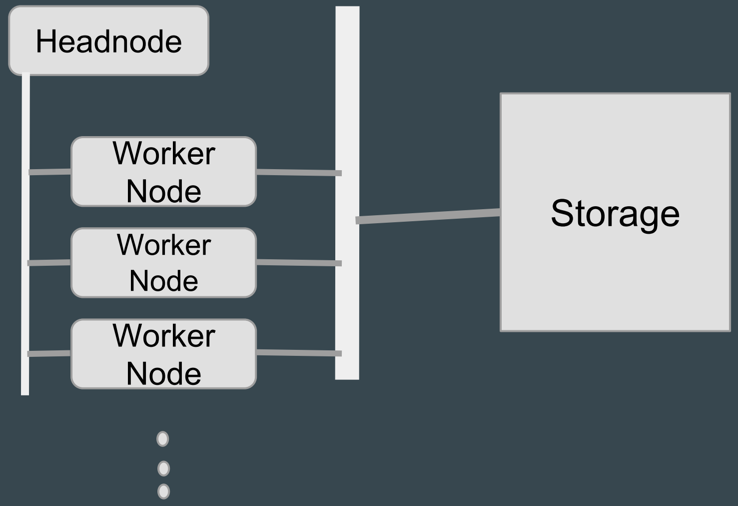
Independent internal to the task and its individual data flow

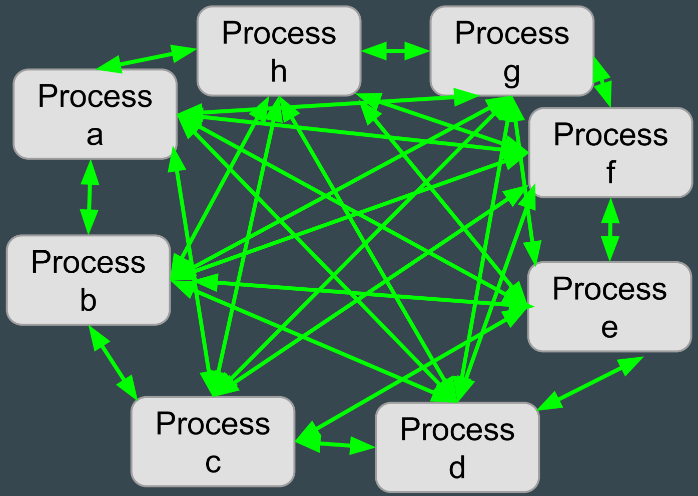
Special setup can improve speed but is not strictly required

Types of HPC

More Interconnects

Tightly coupled workflows



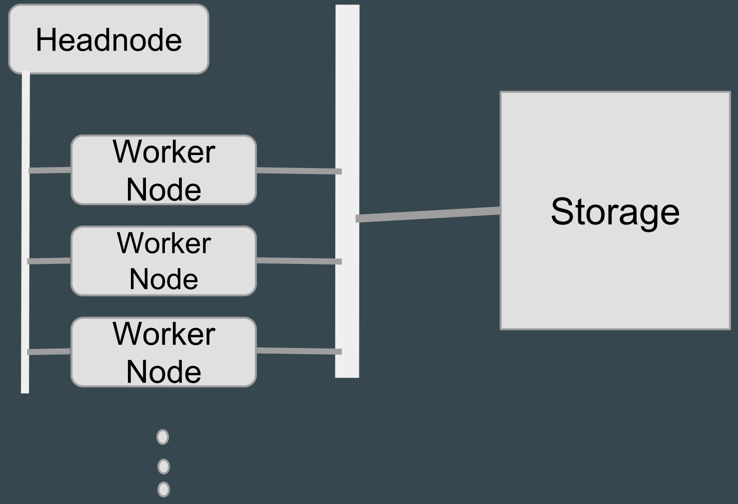


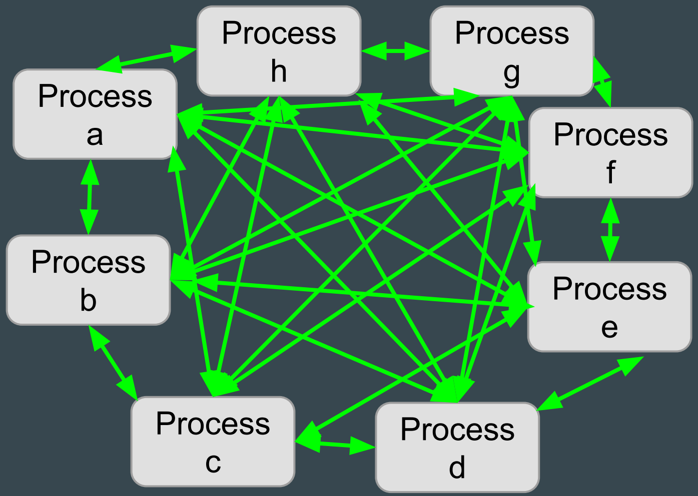
Tightly coupled jobs have interdependent processes, and Shared memory.

Types of HPC

More Interconnects

Tightly coupled workflows





Tightly coupled jobs have interdependent processes, and Shared memory.

Requires special physical set up and programing (PVM or MPI)

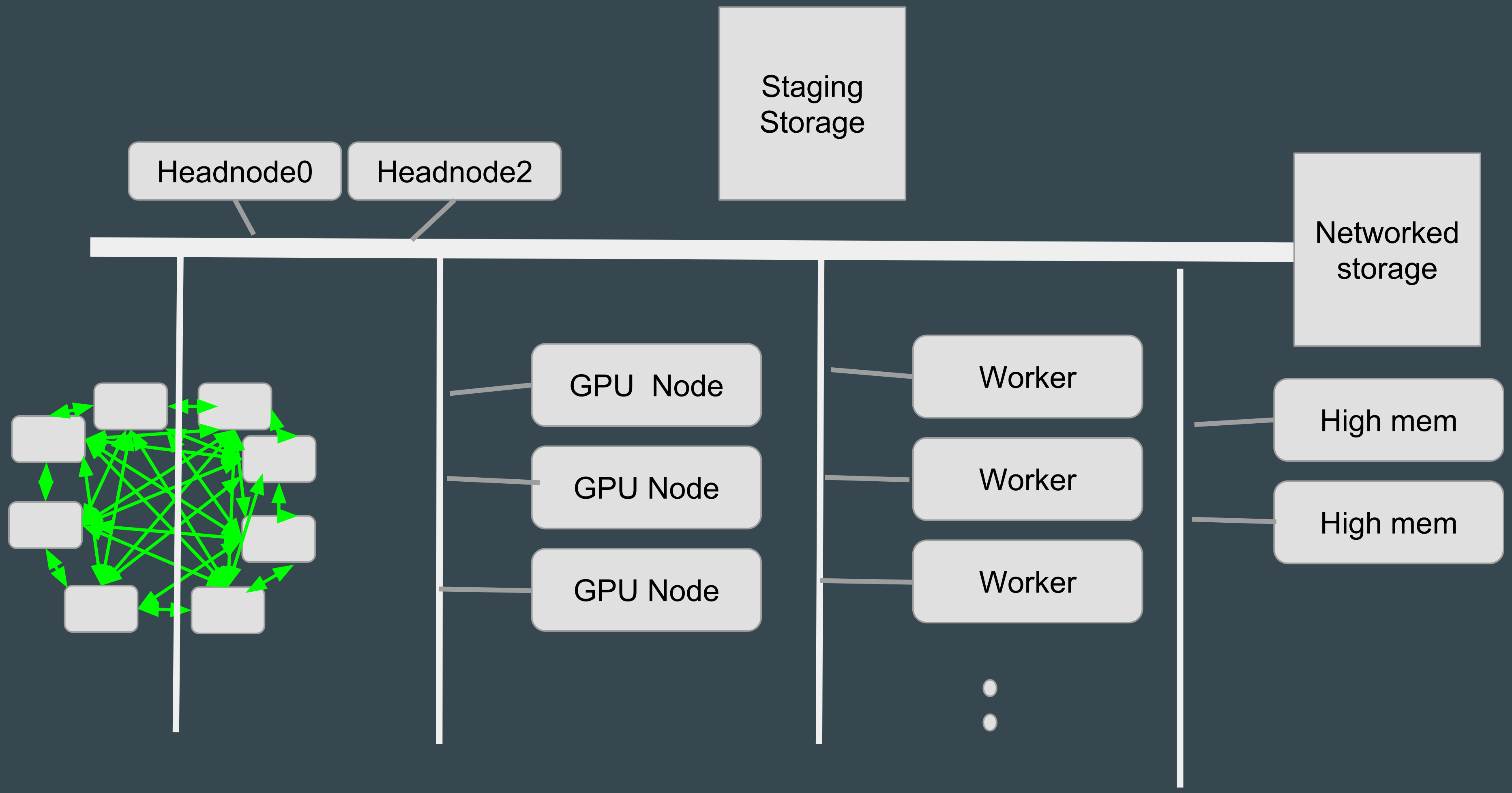
Types of HPC

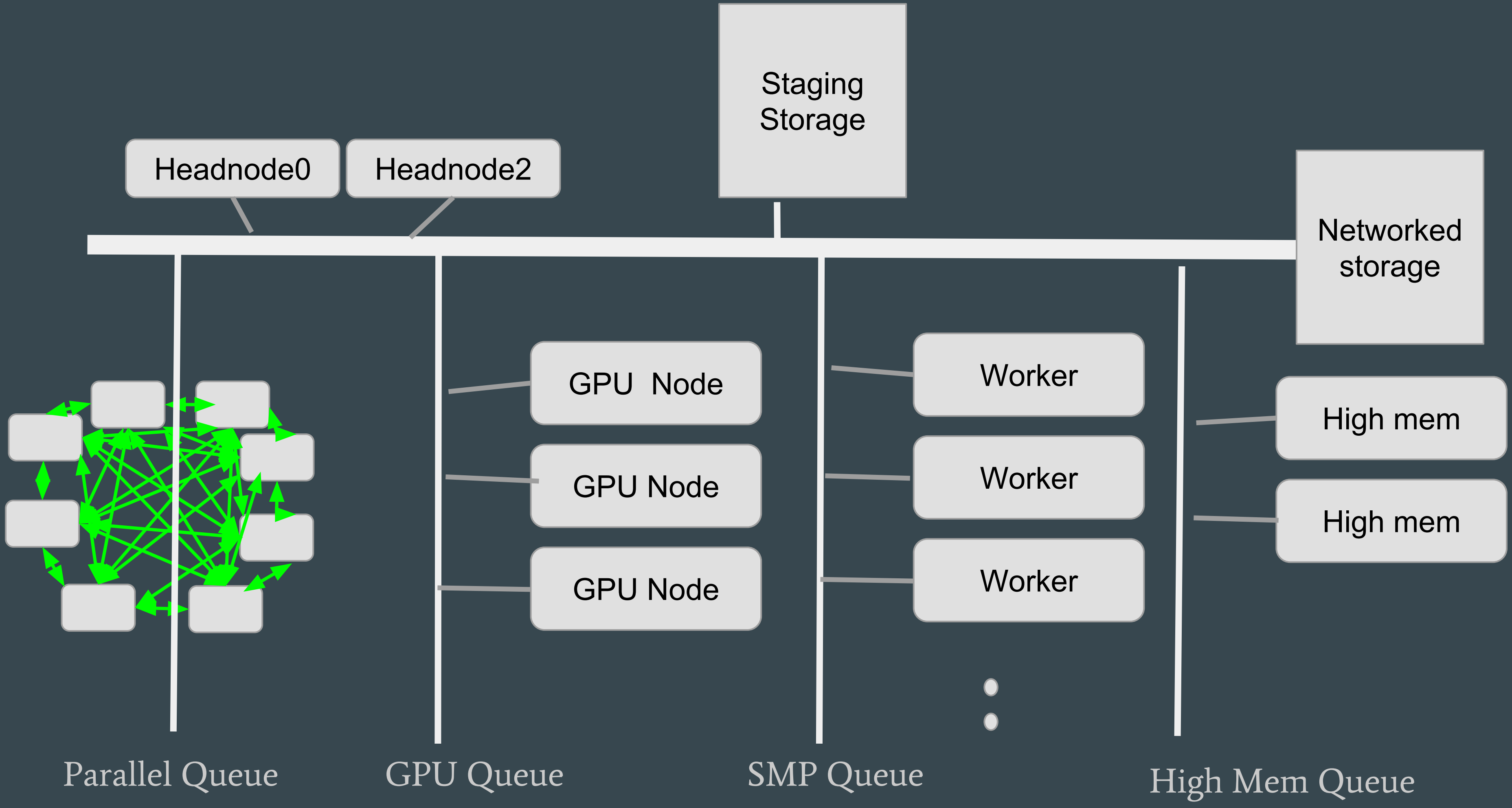
Acceleration

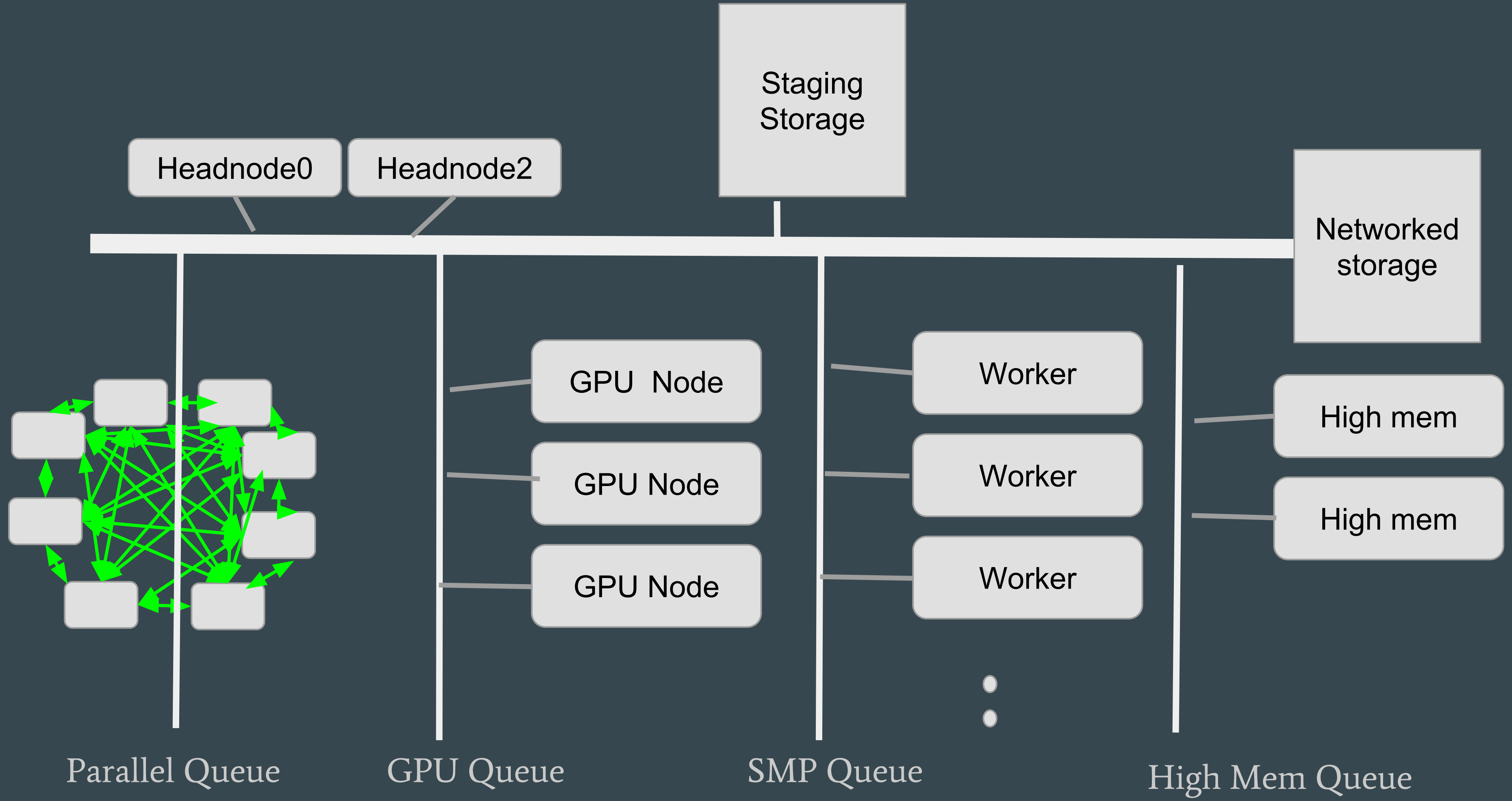
GPU’s: cssentially a dense tightly bound system.

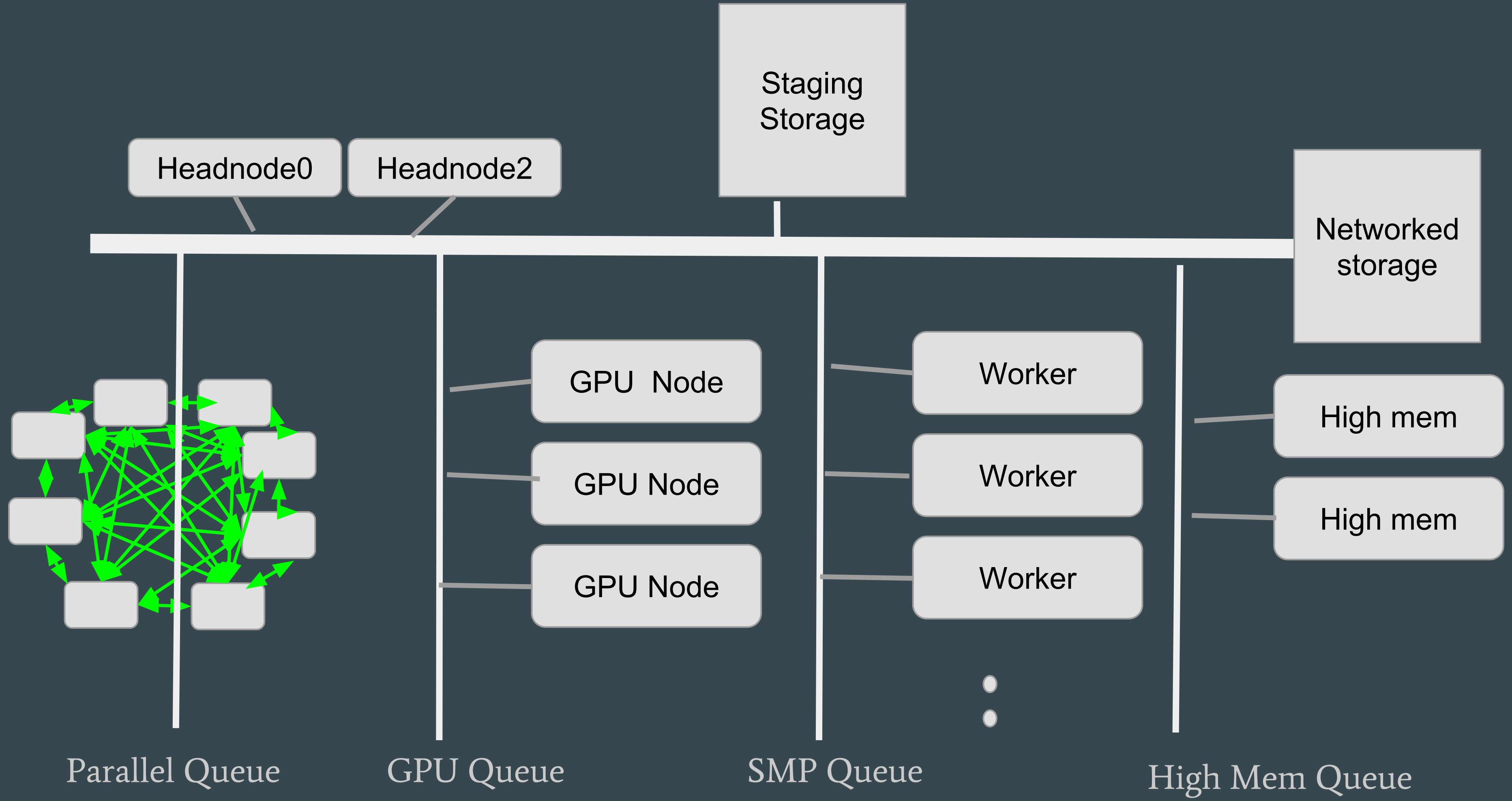
ARM: Single Instruction Multiple Data (SIMD) vector processors

FPGA: Field Programmable Gate arrays Customizing the physical structure of the chip for specific problem sets.


Staging Storage
Headnode0
Headnode2
Networked storage
GPU Node
GPU Node
GPU Node
Worker
Worker
Worker
High mem
High mem


Staging Storage
Headnode0
Headnode2
Networked storage
GPU Node
GPU Node
GPU Node
Worker
Worker
Worker
High mem
High mem
Parallel Queue
GPU Queue
SMP Queue
High Mem Queue


Staging Storage
Headnode0
Headnode2
Networked storage
GPU Node
GPU Node
GPU Node
Worker
Worker
Worker
High mem
High mem
Parallel Queue
GPU Queue
SMP Queue
High Mem Queue


Staging Storage
Headnode0
Headnode2
Networked storage
GPU Node
GPU Node
GPU Node
Worker
Worker
Worker
High mem
High mem
Parallel Queue
GPU Queue
SMP Queue
High Mem Queue

A mile high View

- Solutions are Organizational
- Matches the hardware with
  - Data flow
  - Computational patterns.

Common Queue Scheduler systems

For distributed work

- HTCondor
- Yarn

For localized systems

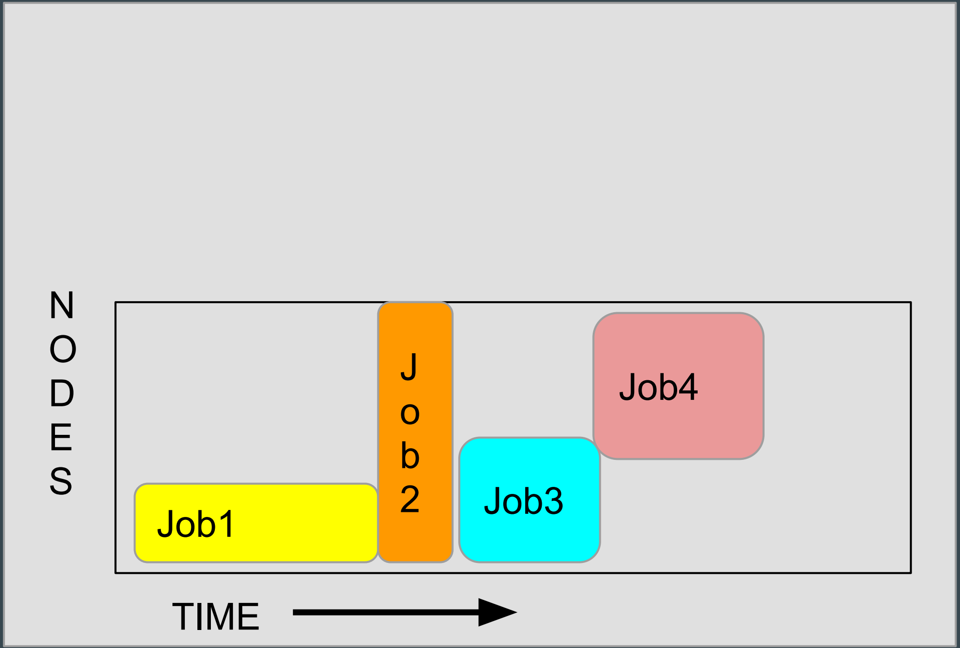
- Grid Engine (SGE, UGE)
- PBS/torque
- Slurm

Queue handling

- First in First Out FIFO
- Priorities (Backfill scheduling)
- Preemptions

Queue handling

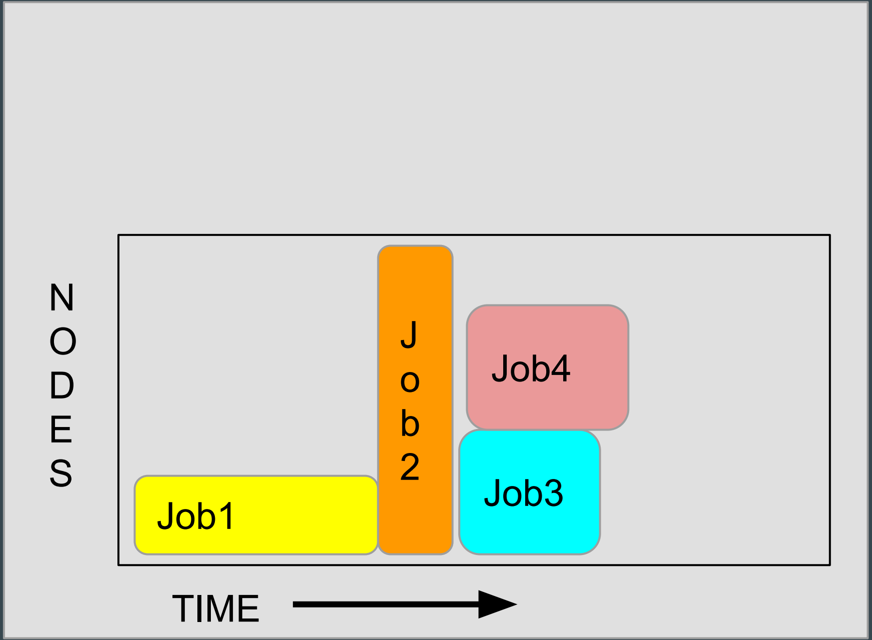
First in First Out   FIFO

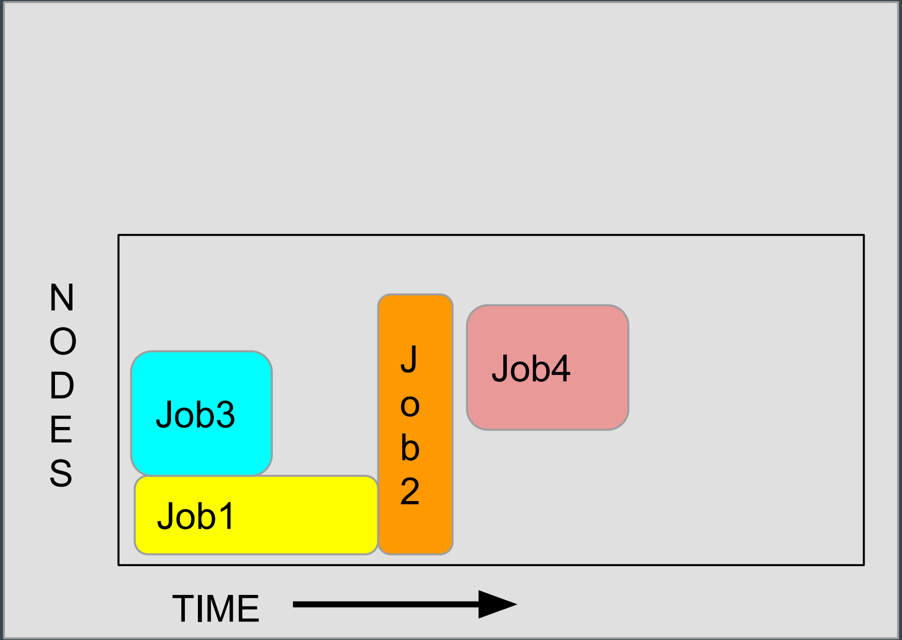


Queue handling

First in First Out FIFO

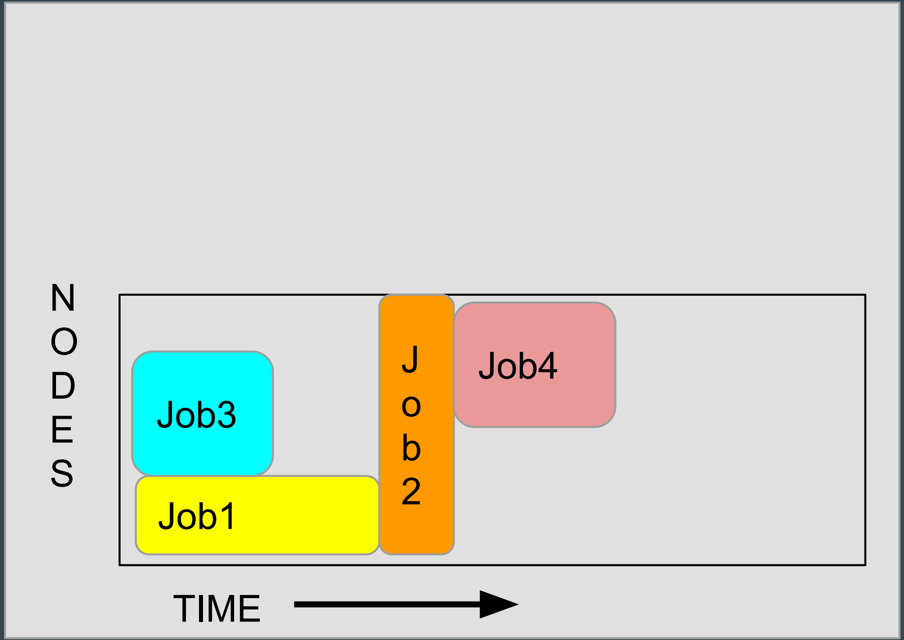
BackFill



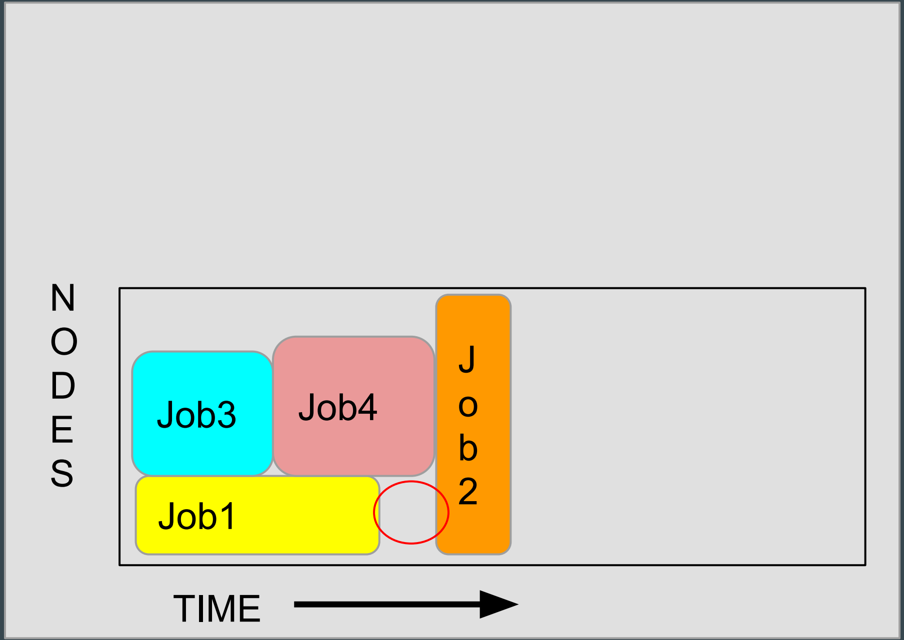


Queue handling

BackFill



Optimizing backfill



Fair share

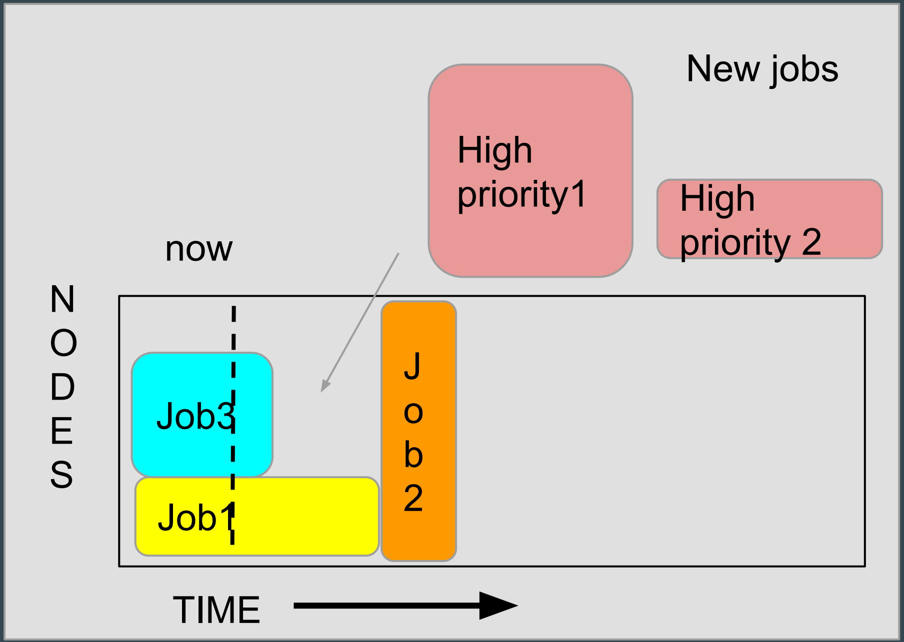
Adds Priorities to backfill scheduling

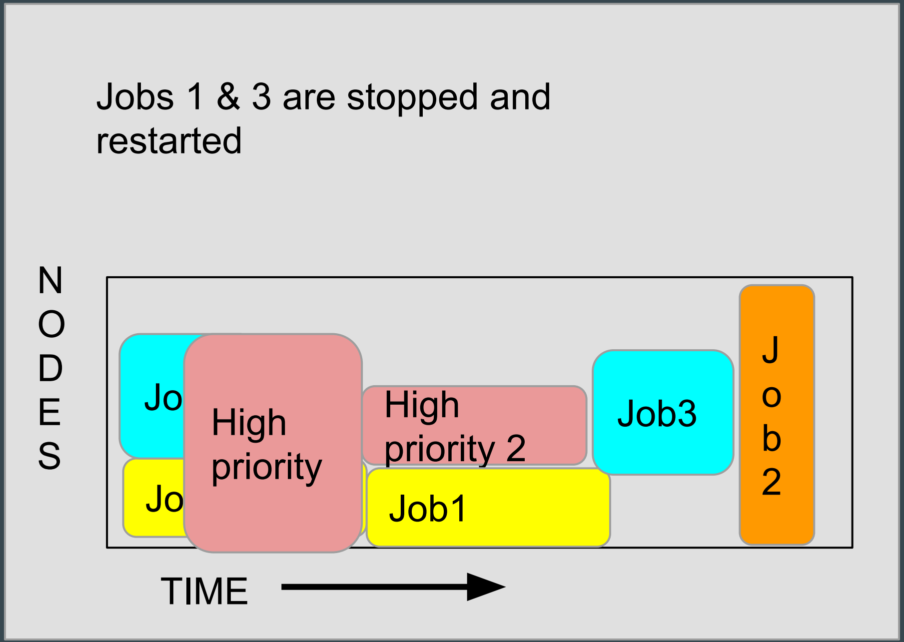
Weight several factors to generate a priority level for each job

- How long it has been waiting
- Job Size
- Where the job is being run
- User’s base priority level

Queue handling

Preemption





What is Checkpointing?

- Saves a representation of the current state at regular intervals
- Allows Automation
  - Scheduler with automatically restarts
  - Add a function at the beginning to check for the save state file
    - If a save state file exists reads it in and starts from there.

```
If statefile exits:
     Startjobat(statefile)
else:
     Startjobat(beginning)
```

When to use Checkpointing

- Long running jobs
    - To not start over if there’s a “hiccup”
        - Balancing between speed and loss
            - Writing to a file is much much slower than anything else.
            - Longer period between checkpoint means faster runs
            - Potential lost time with a restart is bigger.
    - Runtime is longer than maximum job time

Planning workflows

- Seperate jobs
- Identify dependencies
- Decided what (if any) intermediate data to keep
- Match your pattern to available resources and limitations.

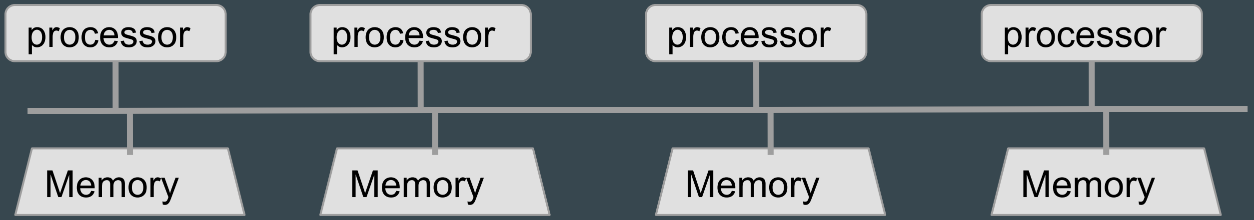
Turning compute-bound problems into I/O-bound problems.

SMP and NUMA

Turning compute-bound problems into I/O-bound problems.

SMP and NUMA

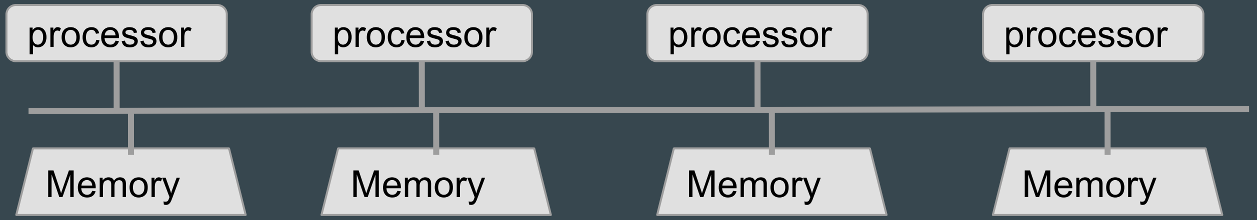
Shared Memory Processing *Uniform access*



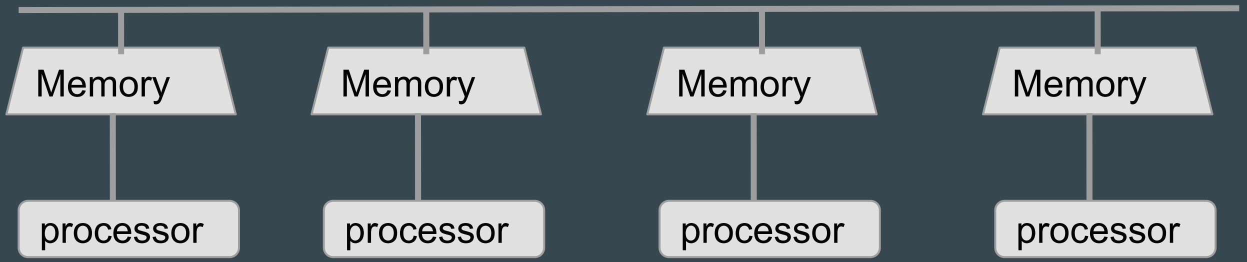
Turning compute-bound problems into I/O-bound problems.

SMP and NUMA

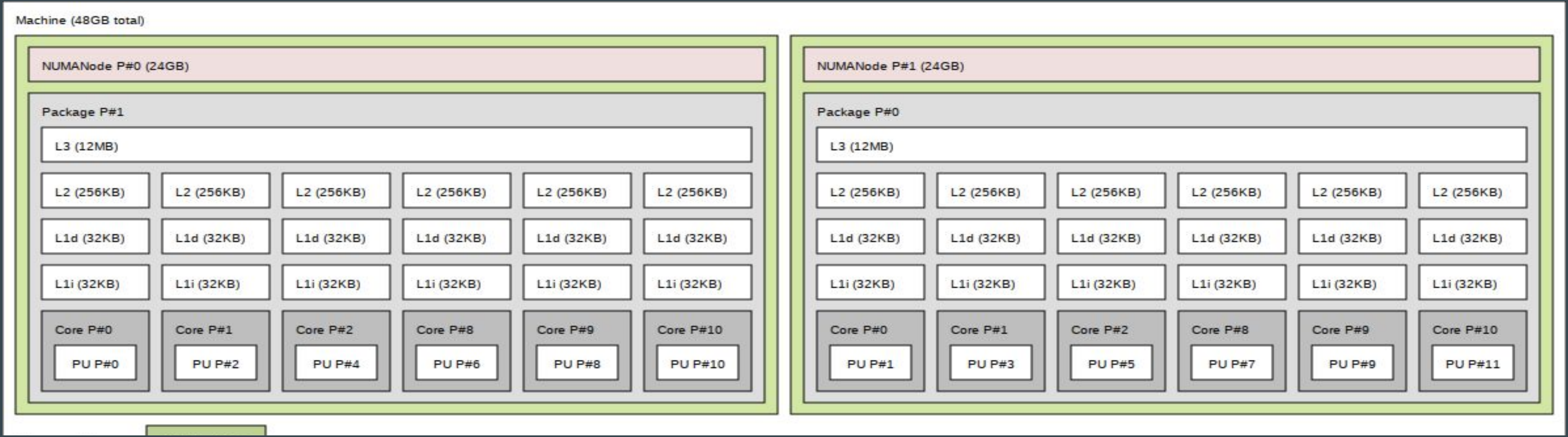
Shared Memory *Uniform access*

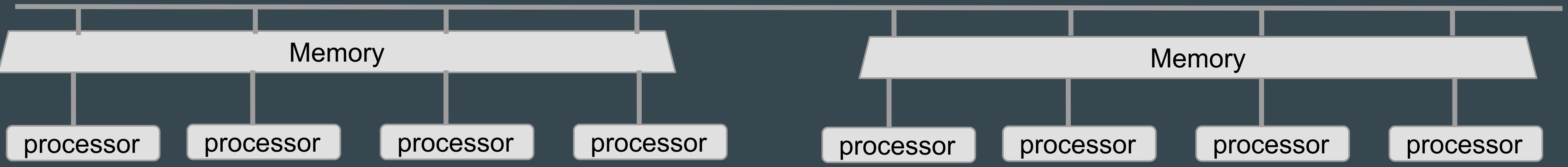


Shared Memory *Non-Uniform Memory Access*

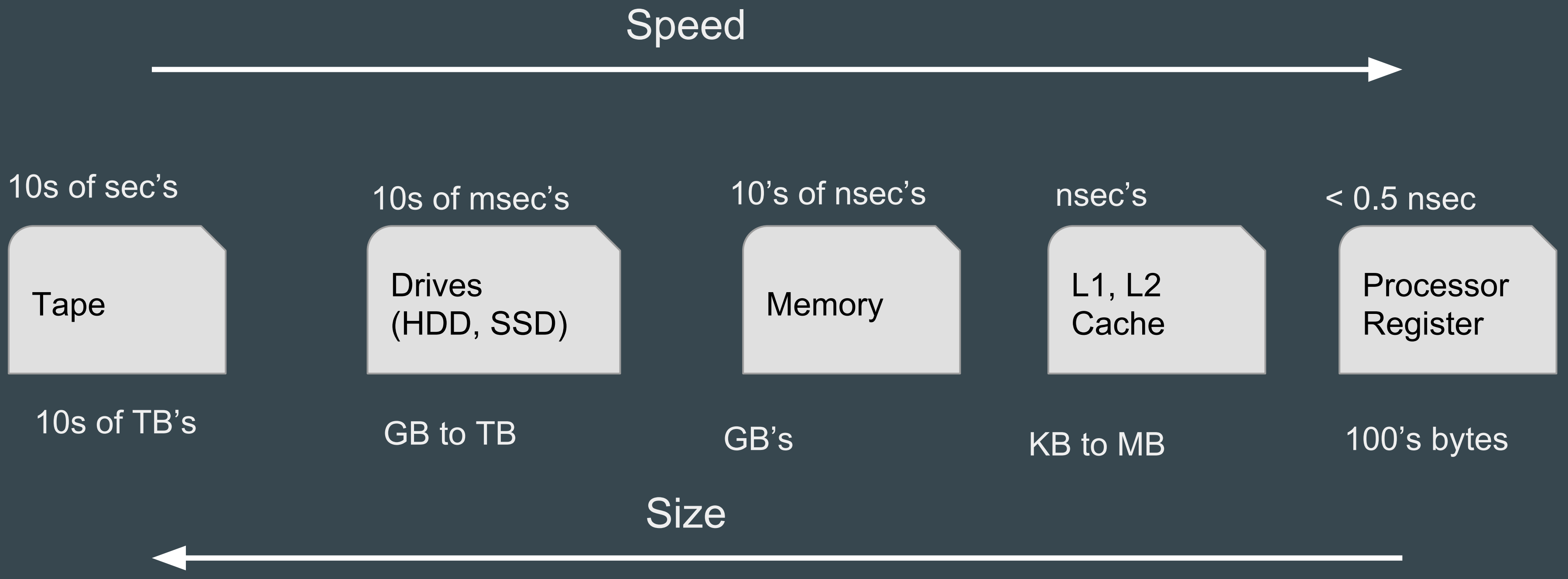


Turning compute-bound problems into I/O-bound problems.
Everything is a graph





Turning compute-bound problems into I/O-bound problems.urning compute-bound problems into I/O-bound problems.



Distance = 1/Speed

Remote networked storage	Slow - not for computing
Nearby networked/warm storage	Pre-staging Completed data
‘Fast scratch’ (or /tmp if not available)	Staging, interprocess holding, bigger than RAM for for worker I/o
Tmpfs (memory as filesystem)	For faster I/o

Distance = 1/Speed

Caution every system is different

Remote networked storage	Slow - not for computing
Nearby networked/warm storage	Pre-staging Completed data
‘Fast scratch’ (or /tmp if not available)	Staging, interprocess holding, bigger than RAM for for worker I/o
Tmpfs (memory as filesystem)	For faster I/o

Check with your local systems documentation

Or ask the system folks.

Modules

Modules

modules allow us to keep the working space clean, when there are a lot of installed programs.

$module avail
Will list the currently available modules

$module load <*modulename*>
Will make the module available
Be explicit!

$module unload <*modulename*>
Will unload the module

$module purge
Will unload all modules

$module whatis <*modulename*>
Will show the module information

Examples

SGE basics

```
#!/bin/bash
# bigio_example.sh

# jobname
#$ -N example

# use this shell
#$ -S /bin/bash

# how much memory per processor
#$ -l h_vmem=1g

# maximum run time
#$ -l h_rt=00:05:00

# parallel environment SMP, 4 processes
#$ -pe smp 4

# email user when done
#$ -m me@myemail.com

runscript “$mytmp”
```

“Big” I/O

a. mkdir mytemp @
   i. /tmp, or /dev/shm
b. Copy files into and untar
   i. cp remote/myfiles.tar mytempdir/
   ii. tar xf myfiles.tar
c. Run jobscript
   i. with working output pointed to mytempdir
d. Tar contents of mytempdir
   i. tar cf myresults.tar myfiles
e. Copy the results back to storage
   i. cp myresults.tar remote/myresults
f. (check the move with checksum)
g. Cleanup (some systems do this for you)
   i. rm -r mytempdir

SGE
Using the /tmp space

```
#!/bin/bash
#$ -N bigIO_example
#$ -S /bin/bash
#$ -l h_vmem=1g
#$ -l h_rt=02:00:00
#$ -pe smp 4
```startupscript
# mydir="$2" tmploc="$1"
starttar=$3 #
mytmp="$1"/"$2"

mkdir "$mytmp"
cp "$starttar" "$mytmp"
cd $mytmp
tar xf "$startar"

runscript "$mytmp"

tar cf results.tar $mytmp
cp results.tar <remote system>
rm -r "$mytemp"

```
```

SGE
Array jobs

```
#!/bin/bash
#$ -N bigIO_example
#$ -S /bin/bash
#$ -l h_vmem=500m
#$ -l h_rt=00:015:00
#$ -pe smp 1
#$ -t 1-5

e

# don't do this
```

SGE
Array jobs

1. Split the files into batched directories
2. Make a file listing those directories
   a. ls -1 * > list.txt
3. Get a count of files
   a. wc -l list.txt
4. Use sed or awk to pop from list.txt and use bigIO pattern

```
#!/bin/bash
#$ -N bigIO_example
#$ -S /bin/bash
#$ -l h_vmem=500m
#$ -l h_rt=00:015:00
#$ -pe smp 1
#$ -t 1-<number from “wc -l”>

infile=$(sed -n -e "$SGE_TASK_ID p" list.txt)

starttar=”$infile”
mytmp=/tmp/”$2”

mkdir “$mytmp”
cp “$starttar” “$mytmp”
cd $mytmp
tar xf ”$startar”

runscript “$mytmp”

tar cf “$infile”_results.tar $mytmp
cp “$infile”_results.tar <remote system>
rm -r “$mytemp”
```

HPC resources

- With ITsupport at Einstein
- XSEDE
- Open Science Grid
- “The Cloud”
    - Amazon, Google, Microsoft and others