Using a cluster effectively
Scheduling and Job Management

Log into the cluster

Go the the etherpad

https://pad.carpentries.org/uofa-sept-2019

Download answers.tar and extract it

Scheduling and Job Management 1

Using a cluster effectively

Presentation contents

Scheduling Theory

Basic Job submission

Parallel computing and Job submission

Distributing resources without scheduling exercise.

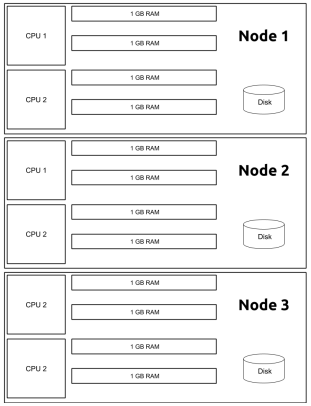
In this example we will be dividing cluster for the entire year, not via batch scheduler

There is more than one right answer.

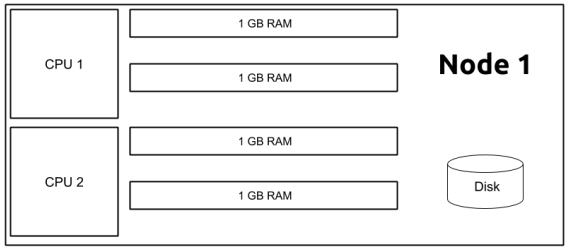
You will be give a sheet of paper representing a cluster and a ziplock bag with colored squares and rectangles.

The color of the shape identifies to whom it belongs.

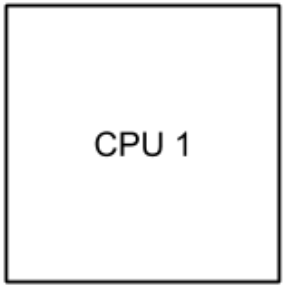
Diagram of Materials



Cluster



Node



Core

RAM

Distributing Resources

You are appointed to run a HPC facility in a fictional nation (CW).

You have a number of limited resources 3 nodes with 2 cores each.

You also have requests from number of users with compute needs.

Distributing Resources

Users need to guarantee that they will actually get any allocation that they are given.

Discus among group: Users need to request resources what questions do you ask?

Please allocate the requested resources (cores) in this instance by putting the shapes onto the sheet of paper representing the cluster.

Please use the representative color for each of the users jobs.

Look at your colleagues allocations and discuss the choices made and why.

Needs				
Name	Color	Priority	Jobs	Cores per Job
T'Challa	**Black**	Highest	2	1
Steve	**Blue**	High	1	1
Tony	**Orange**	Med+	2	1
Bruce	**Green**	Med	1	1
Stephen	**Purple**	Low	1	1

Distributing Resources part 2

You are appointed to run a HPC facility in a fictional nation (CW).

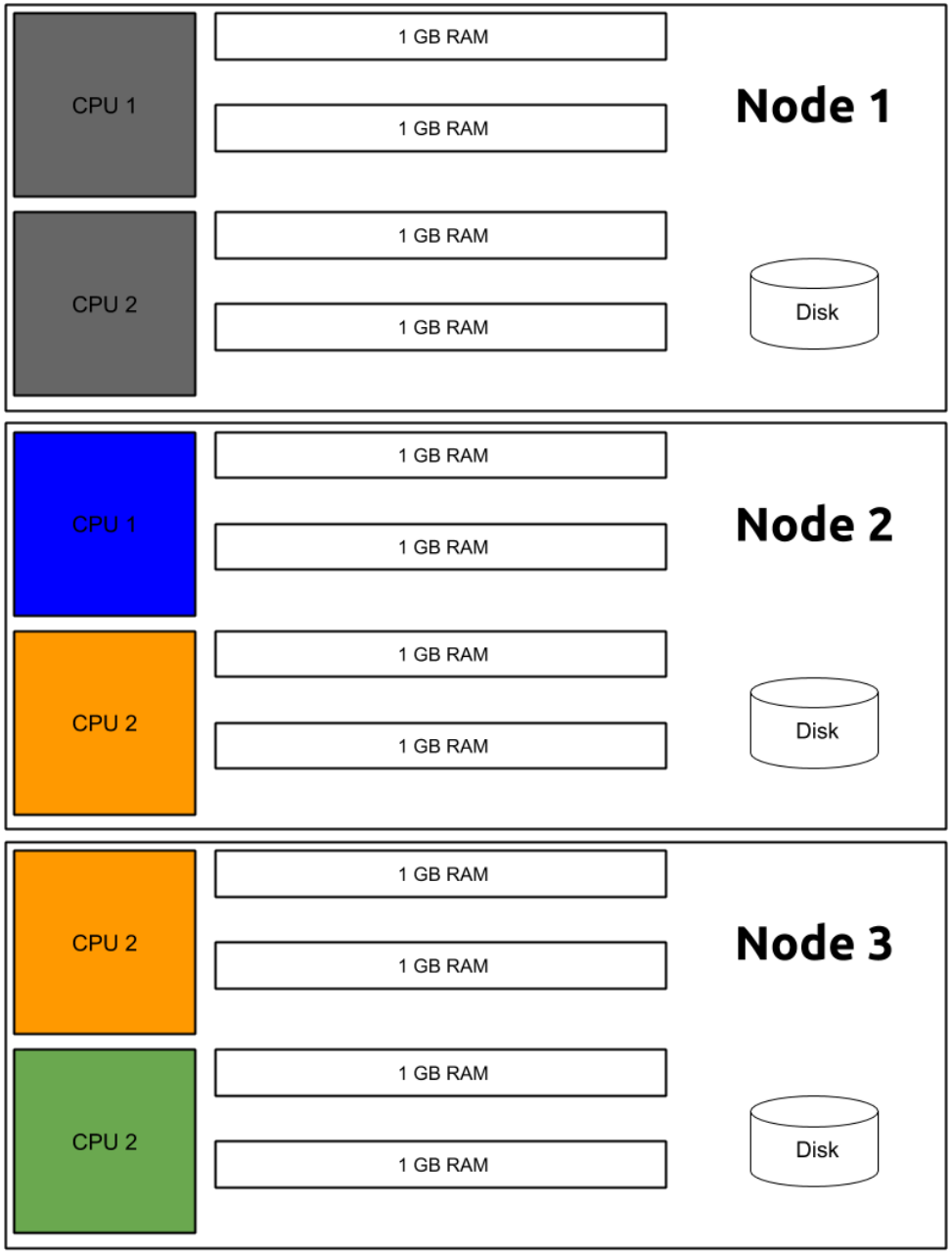
You have a number of limited resources 3 nodes with 2 cores and **4 GB RAM each**.

You also have requests from number of users with compute needs.

Users need to guarantee that they will actually get any allocation that they are given.

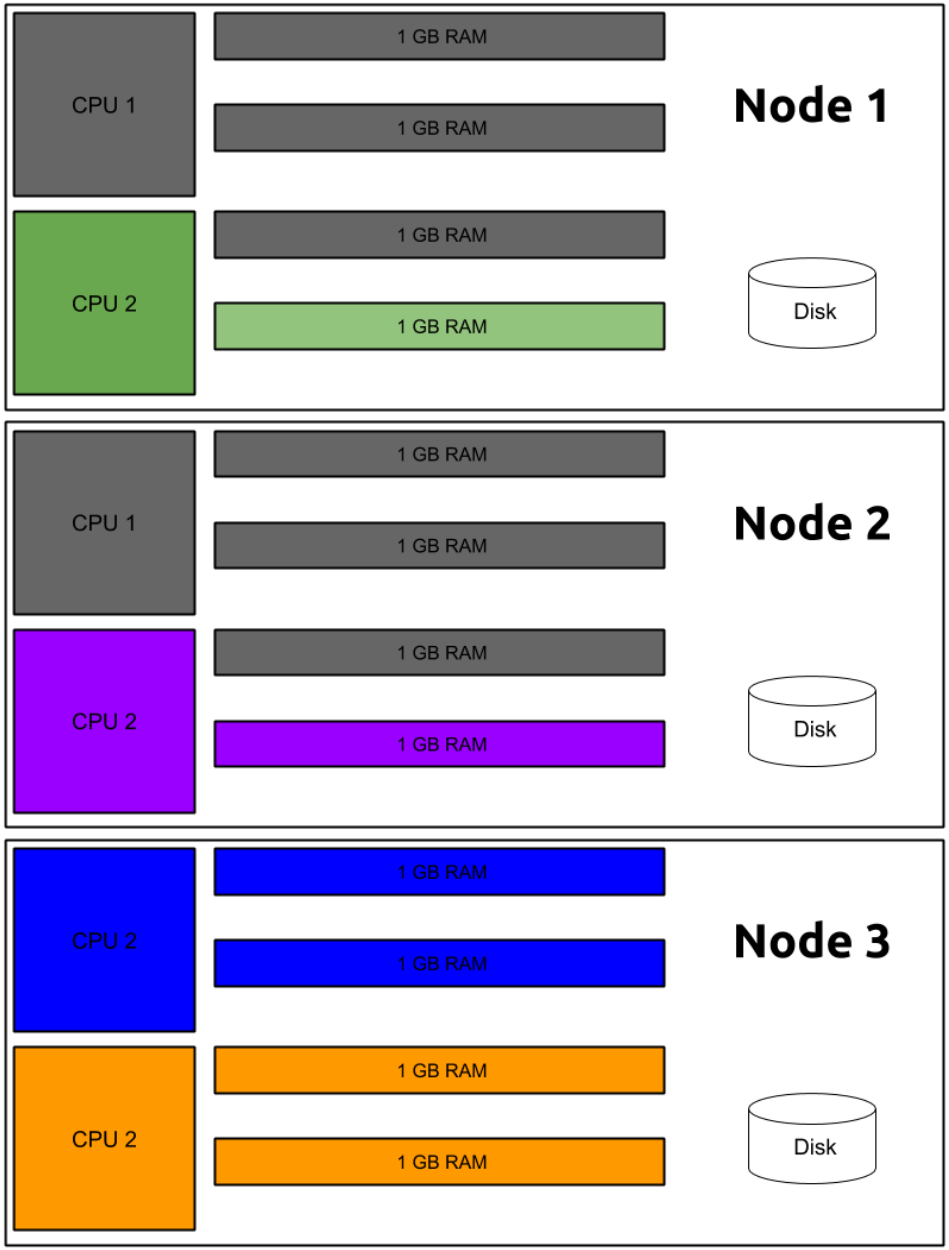
Users need to request resources what questions do you ask?

Needs					
Name	Colour	Priority	Jobs	Cores per Job	**Mem/ core GB**
T'Challa	**Black**	Highest	2	1	3
Steve	**Blue**	High	1	1	2
Tony	**Orange**	Med+	2	1	2
Bruce	**Green**	Med	1	1	1
Stephen	**Purple**	Low	1	1	1



Distributing resources

One possible solution



Distributing resources part 2

One possible solution

How do we (CW) take requests for resources?

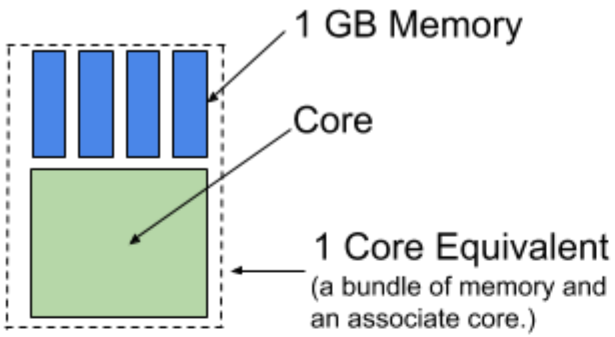
In a Group discuss along yourself:

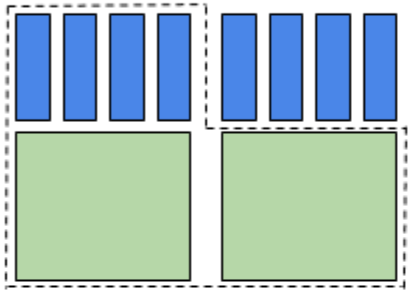
- You have a large number of resources and a large number of request.
- How do you take requests for resources,
  - Remember you have to guarantee that you can have enough resources to give to the people that you have granted allocation.
  - What questions do you ask? What process do you use to make sure you allocate as many high priority users with resources as possible?

Distributing Resources and priority

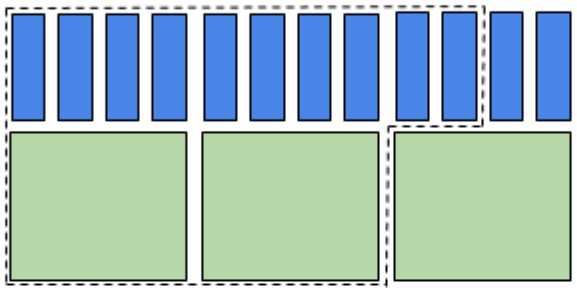
- There are multiple resources cores, memory, GPUs etc…
- In order to be used most resources must be used in conjunction with other ie: A computation that requires a GPU also requires use of some CPU cores and system memory.
- As a consequence priority must be one number. A computation cannot have a RAM priority and a different Core priority, as the priority determines if the computation runs. The use of cores and memory both need to happen and at the same time for the computation to take place.

Core Equivalent





2 Core equivalent used



2.5 Core equivalent used

Compute Canada core equivalent documentation

Core Equivalent

- The Scheduling system has a priority system based upon how many resources used by each group in the past. We need a single number to represent a groups usage as a share of cluster resources.
- Each computation uses a number of resources. We count the usage as the portion of the maximum share of resources of a typical node.
- Ie if a typical cluster node has 32 cores and 128 GB of RAM. Job A uses 1 core and 96 GB RAM (3/4 of RAM on the node) it can be said to use ¾ of a Node, or expressed as the equivalent of ¾ of the cores on one Node. So we can say that Job A uses 32* ¾ = 24 Core Equivalent. We would use this number In determining usage, priority, and allocations.

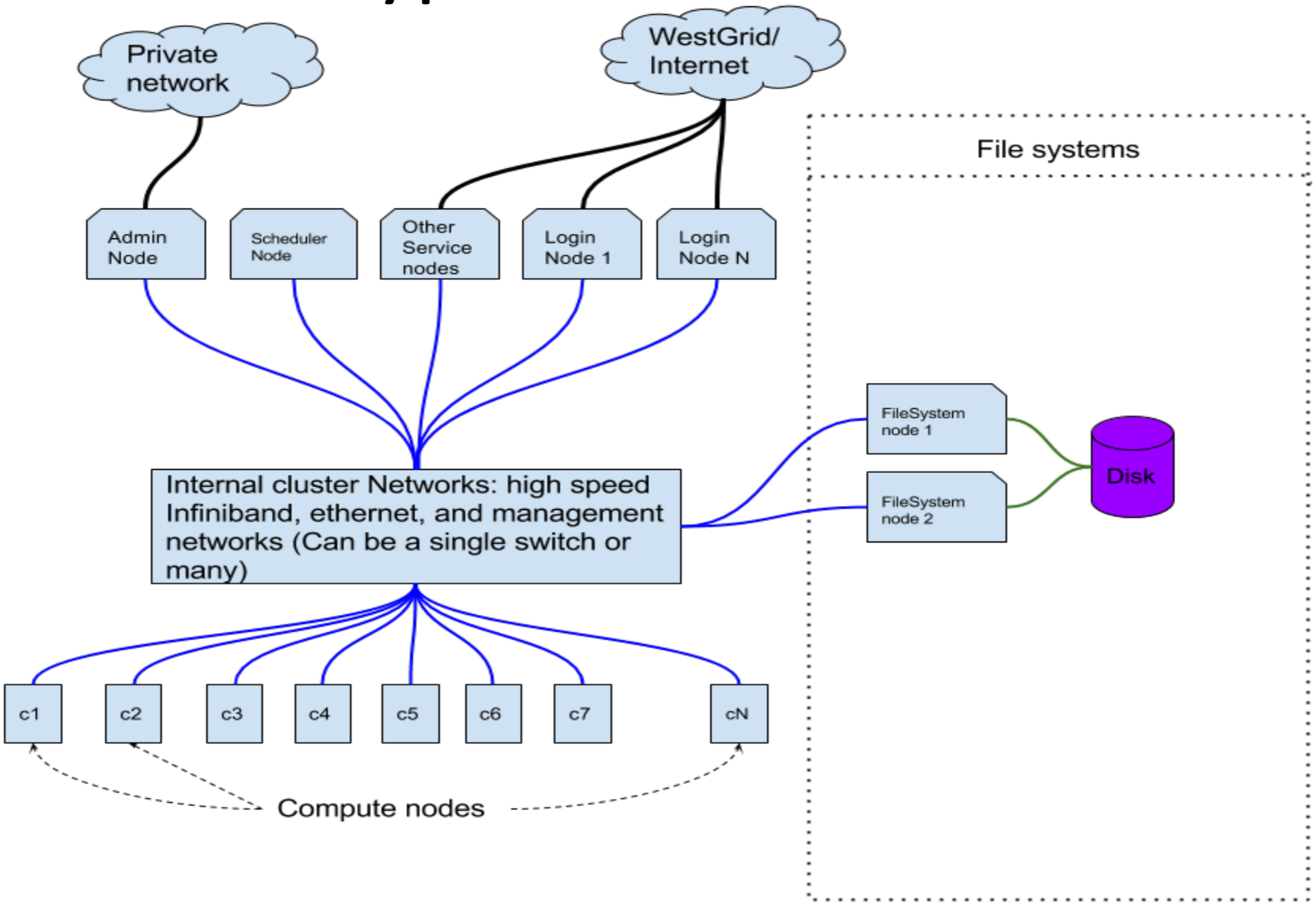
Batch Scheduling

- Is not used when you need a service for example a webserver that runs all the time.
- Is preferred when you have one or more jobs (simulations) that need to be run and you wish to get the results back sometime in the future.
- Your job automatically started by the scheduler when enough resources are available, and you get results back, you may be notified when your job starts and finishes.

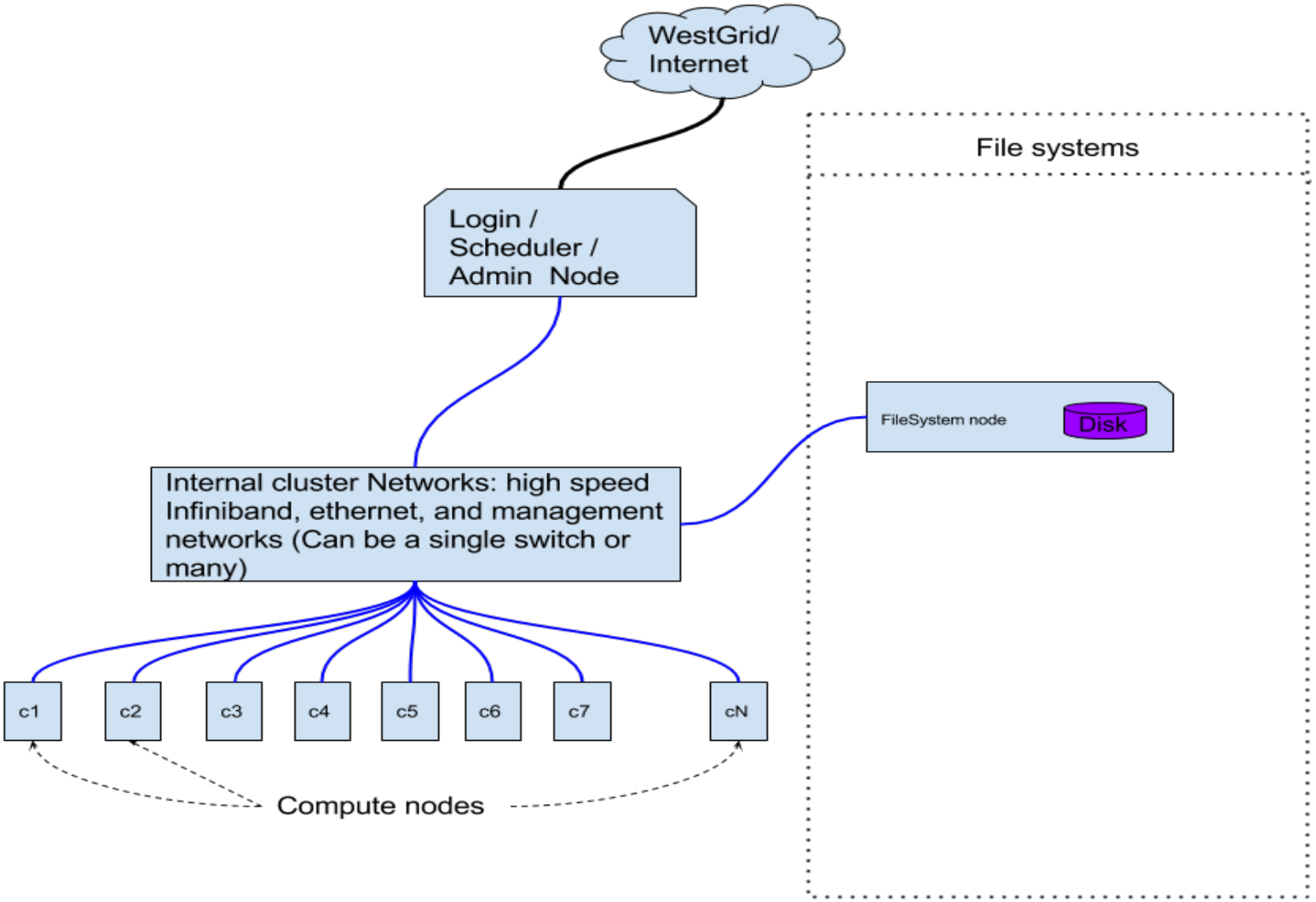
Distributing Resources and batch scheduling

- With batch scheduling a user can submit different sized, length jobs using different amounts of resources.
- All jobs have a priority.
- Any allocation you get is actually a share of the system.

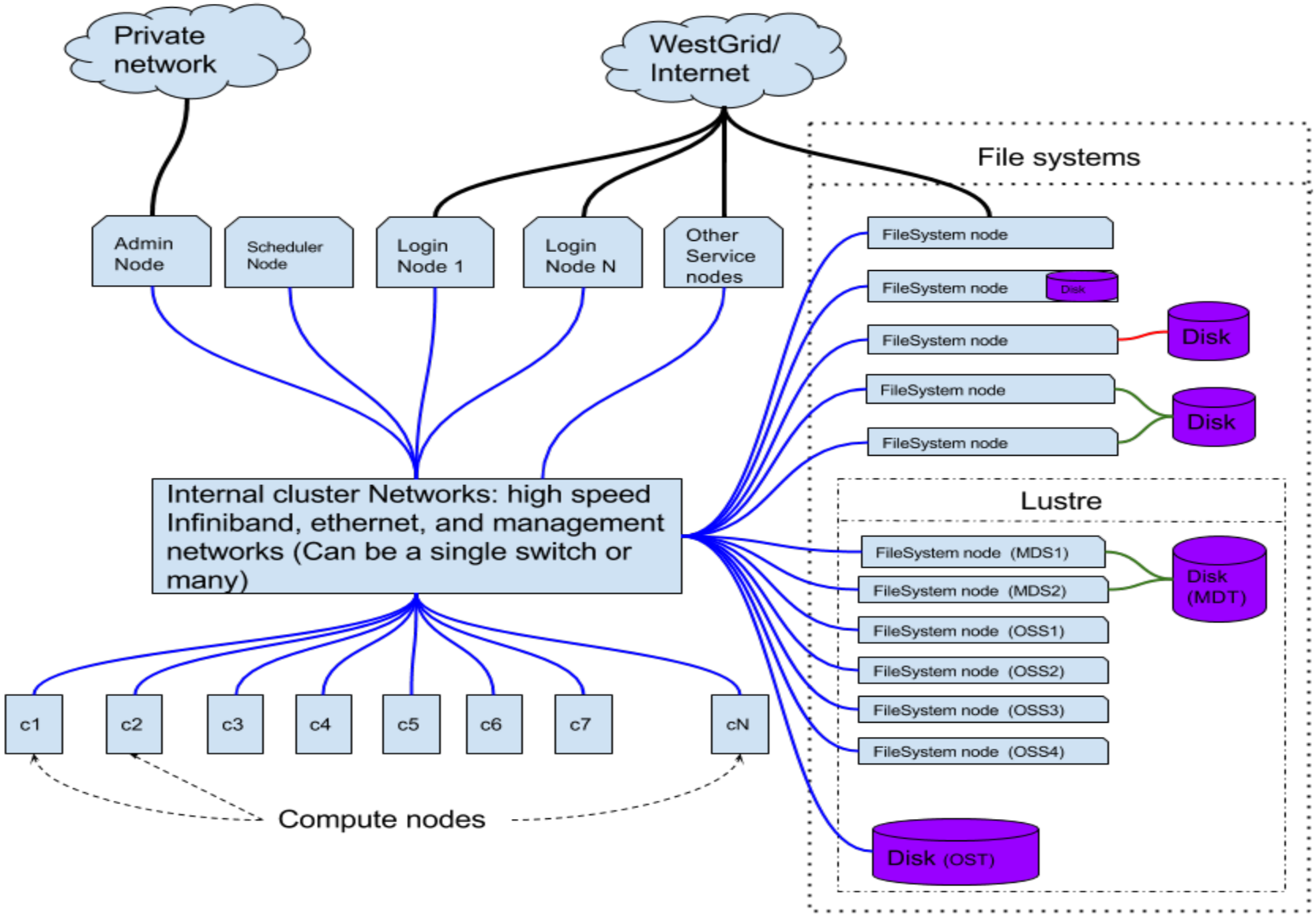
Typical HPC Cluster



Typical small HPC Cluster



Bigger HPC Cluster



Goals of scheduling

- Fairness and policy
- Efficiency / Utilization / Throughput
- Minimize turnaround

Fairness and policy

- Does not necessarily mean everyone or every group gets the same usage.
- An important science project may a get larger allocation.
- Scheduler fairly allocates according to usage policy

Efficiency, Utilization and Throughput

- We want all resources cpus, gpus, memory, disk, software licenses, bandwidth, and more to be all used as much as possible.
- How many gaps are there in scheduling between jobs.

Minimize turnaround

- Goal here is return an answer or result to a user as fast as possible
- Important to users which use iterative process to their goal.
- Minimize time to scientific discovery

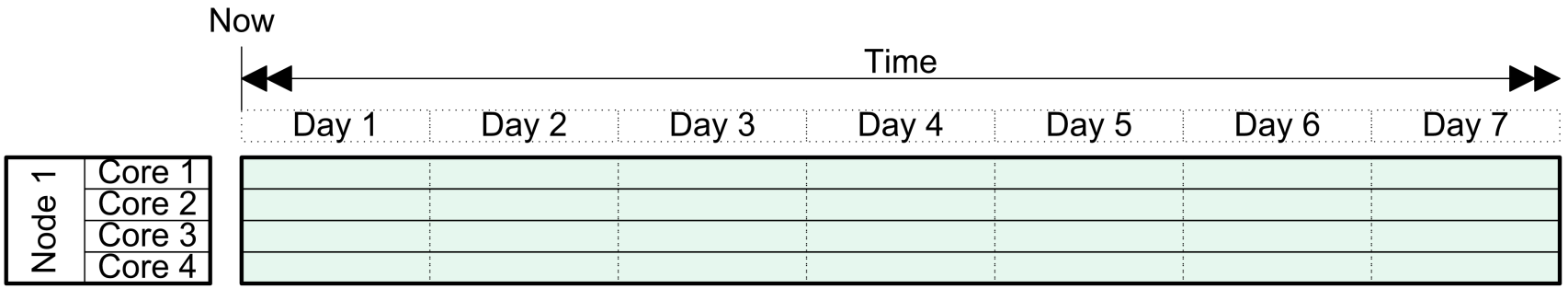
Some insights

- The shorter the walltime which is the maximum time a job will run before being killed, the better we can meet the 3 goals of scheduling.
- Jobs using large amount of resources per job result in a reduction of fairness, efficiency, responsiveness of the scheduling system.
- The more nodes we have the better we can meet these goals.

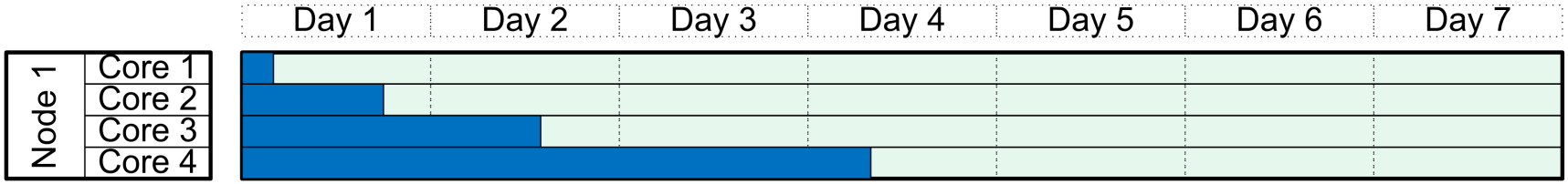
Advantages of Large Clusters

- Larger clusters are more fair, efficient, responsive just by being larger.
- Larger clusters are capable of running larger jobs expanding capability, but if larger jobs are run exclusively we loose the advantage of a large cluster.
- Shared resources such as WestGrid are better and are used more efficiently than multiple small clusters. The larger the scope of shared resources the better.

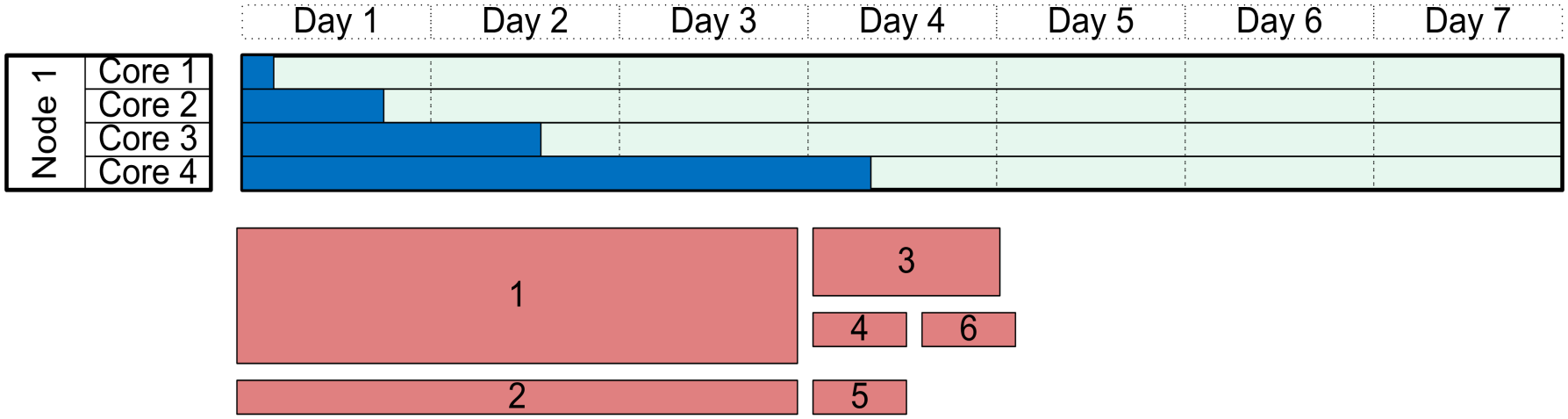
Visualizing single node cluster



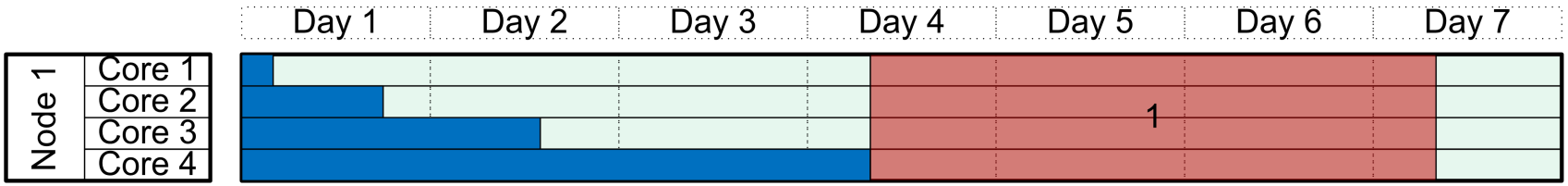
Running jobs



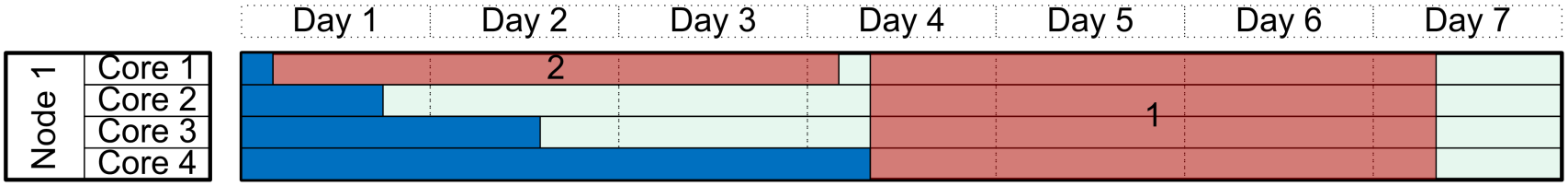
Scheduling jobs in order of priority



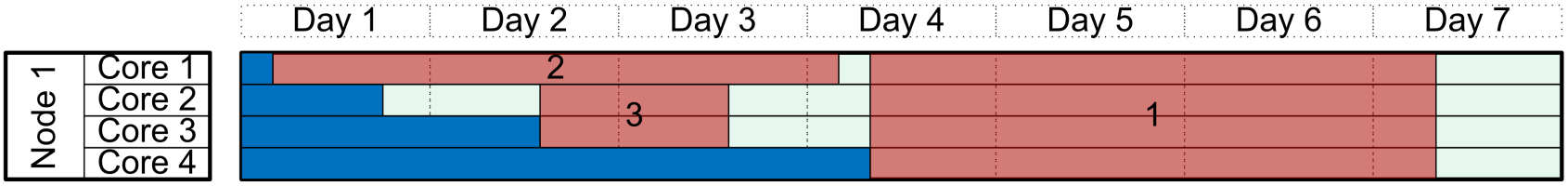
Scheduling jobs in order of priority



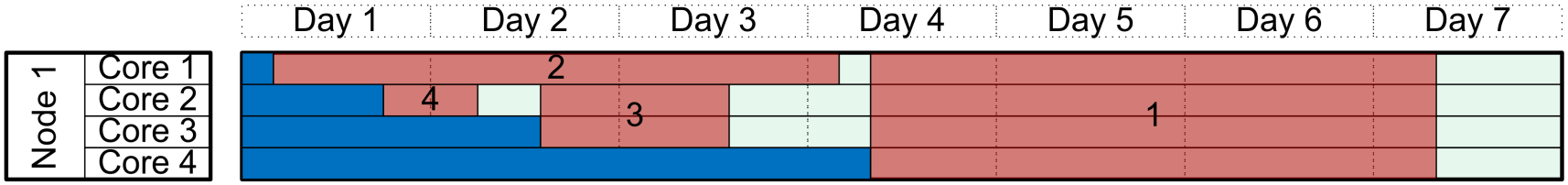
Scheduling jobs in order of priority



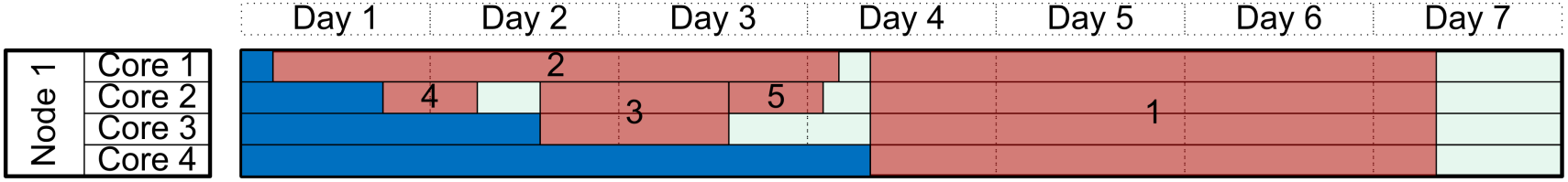
Scheduling jobs in order of priority



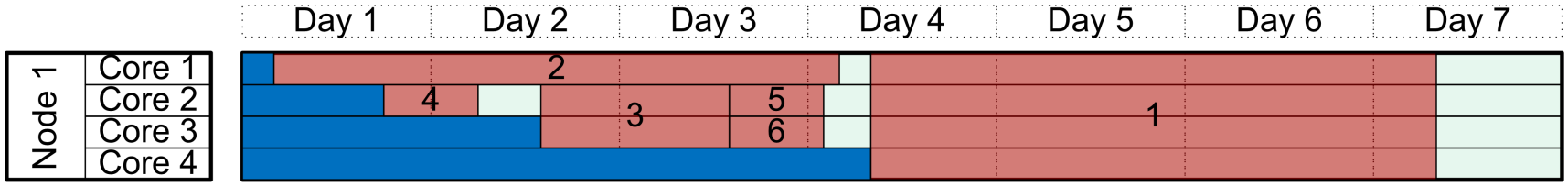
Scheduling jobs in order of priority



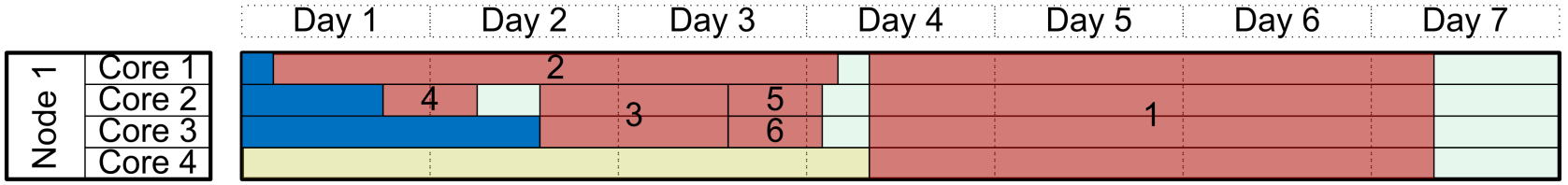
Scheduling jobs in order of priority



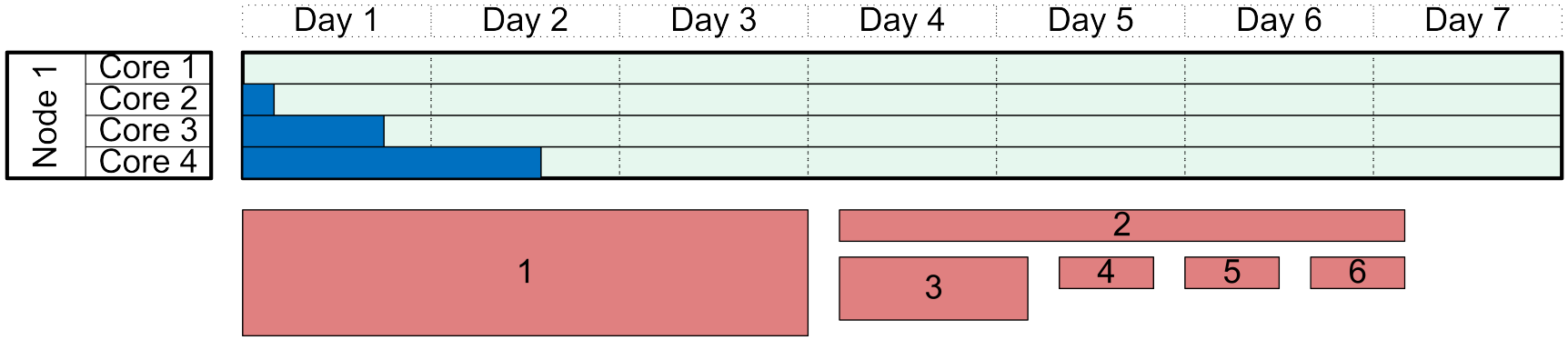
Scheduling jobs in order of priority



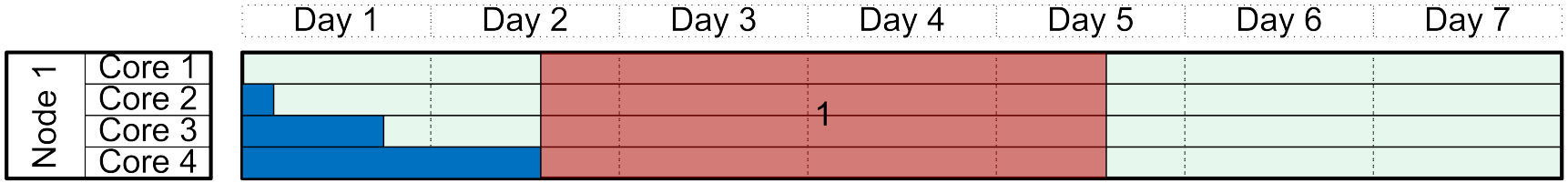
A Job finishes early



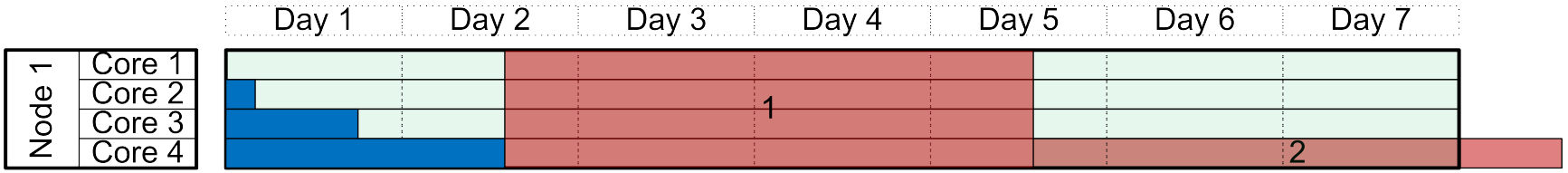
Jobs are rescheduled



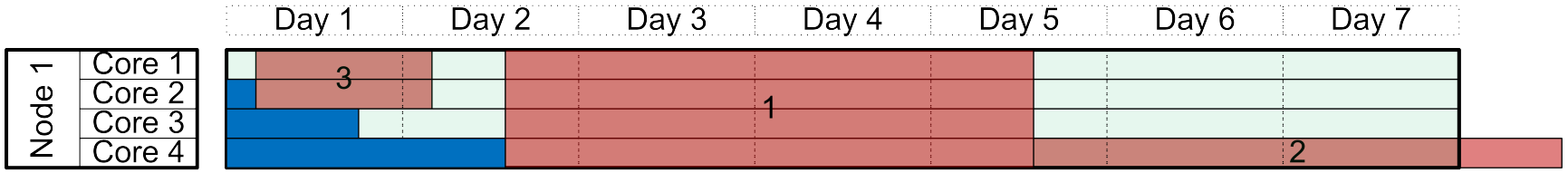
Jobs are rescheduled



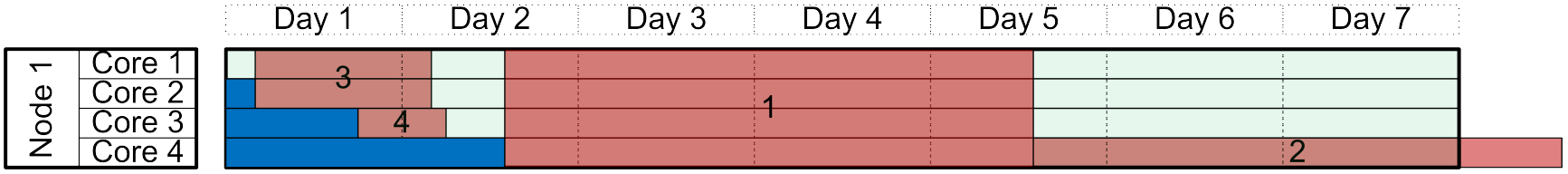
Jobs are rescheduled



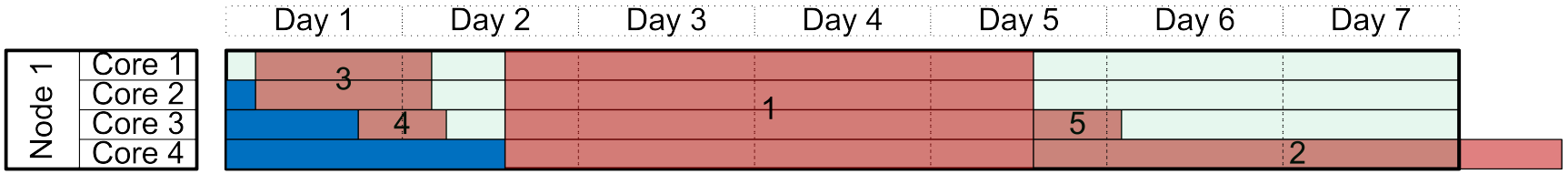
Jobs are rescheduled



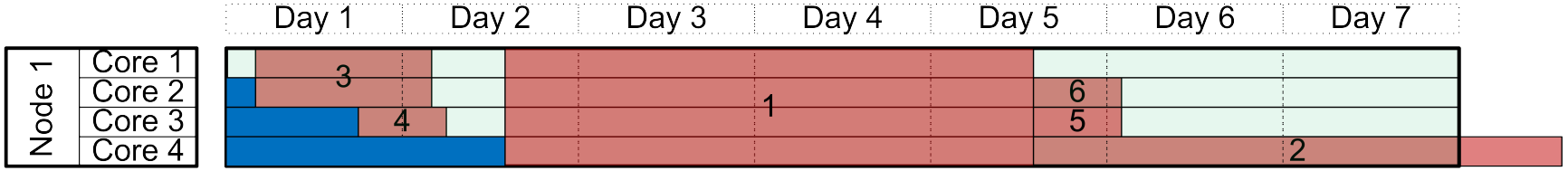
Jobs are rescheduled



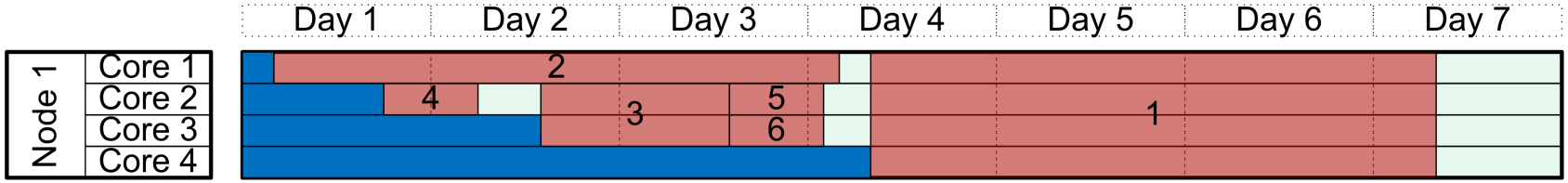
Jobs are rescheduled



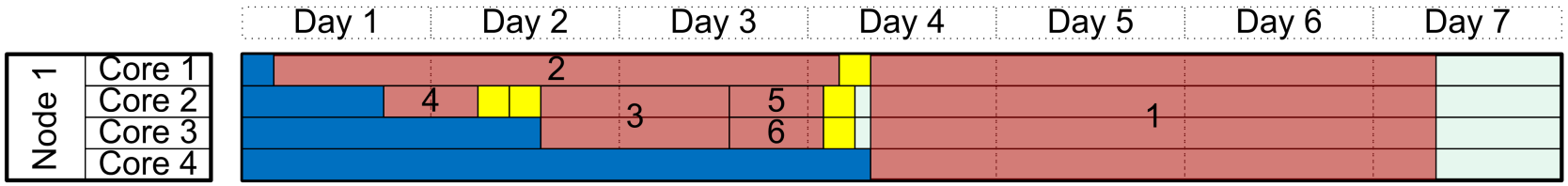
Jobs are rescheduled



Single node cluster



Short serial jobs and Backfill



Myths

If there is a large number of jobs in the queue my job will not run quickly.

- Most of the time these jobs belong to users with very low priority, because they are running a large number of jobs.
- Most of these jobs may not be capable of running as number of running jobs per user may be limited.
- The cluster may have empty processors available for immediate use.
- Deciding if a cluster is busy by number of queued jobs does not work.

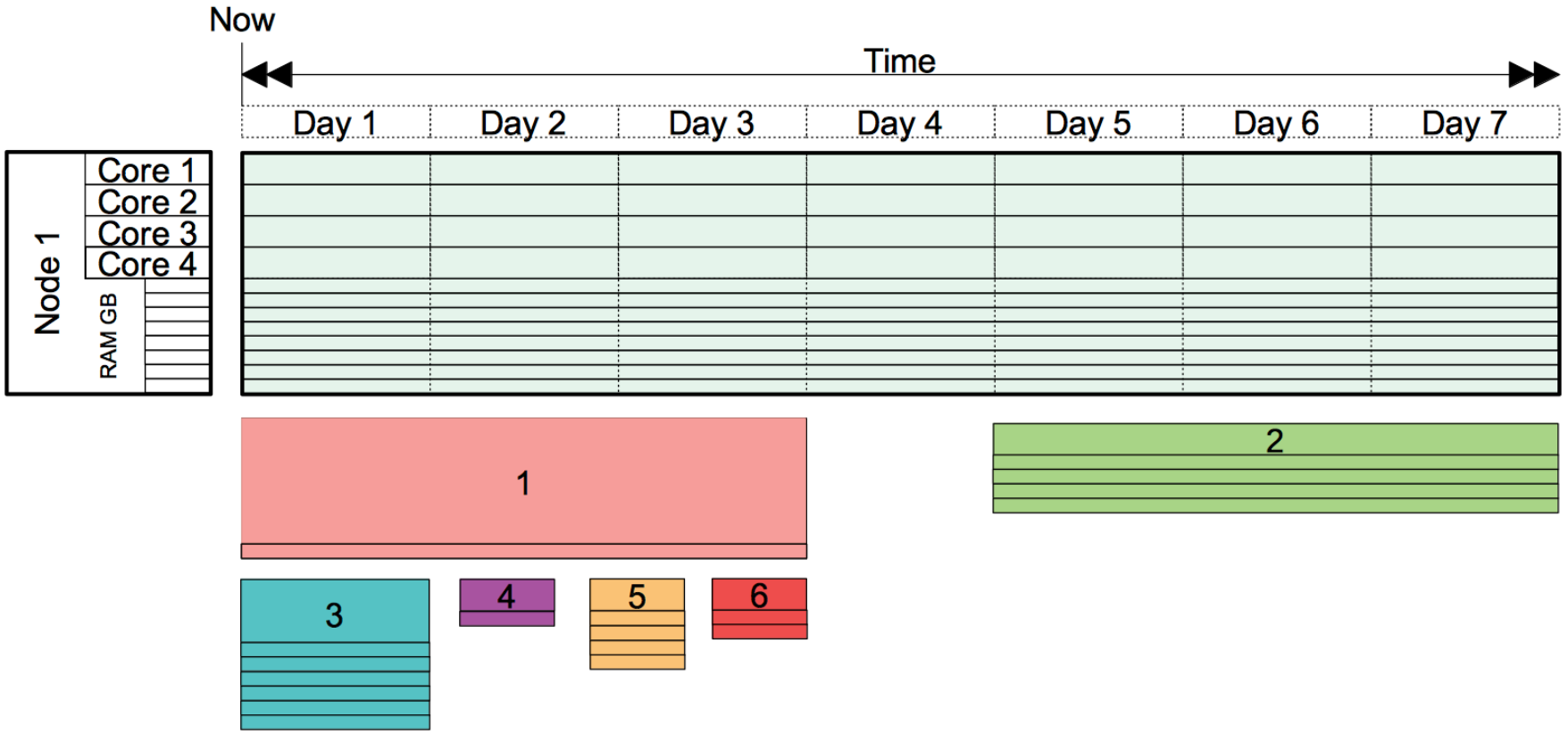
It is better not to submit to many jobs at a time so that other users can run.

- The scheduling system is more efficient if you submit your jobs earlier, as long as you don't go over the usage limits.
- Fairness is insured by the scheduling system.

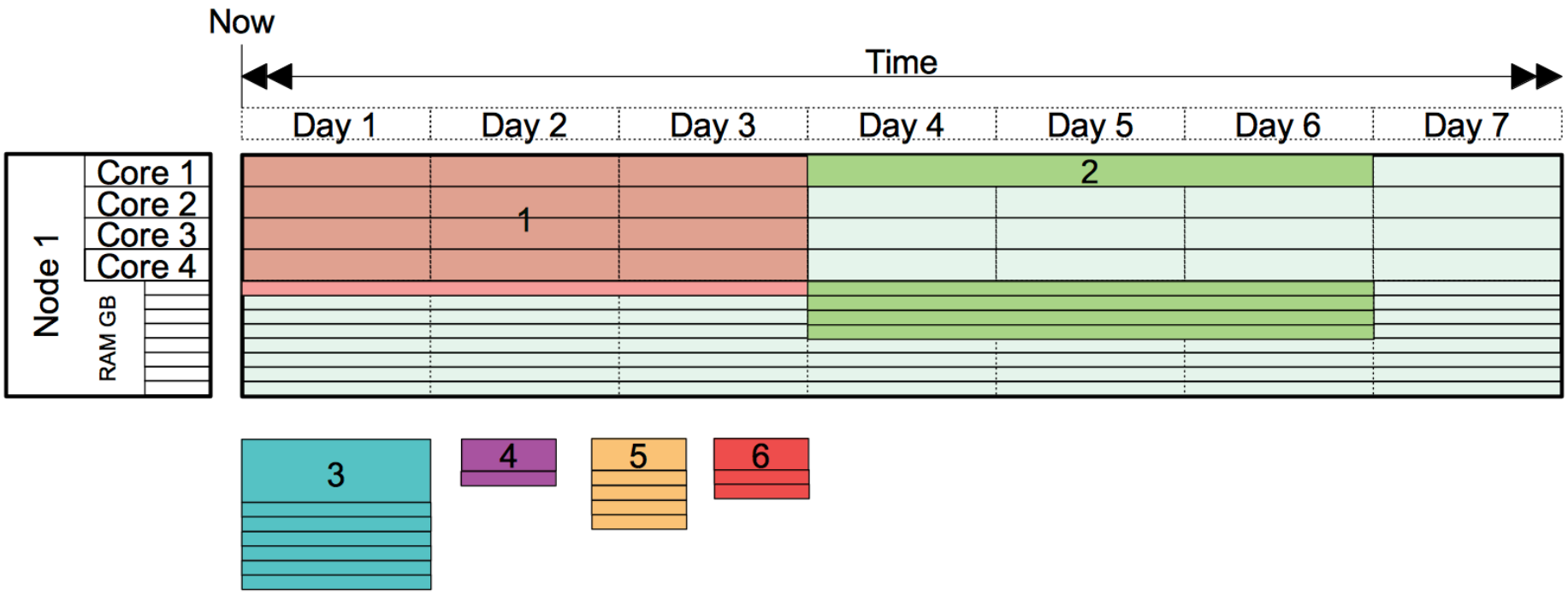
Tips

- Make sure your job can run on the resources available on the cluster.
- Look at the state of cluster/account/Jobs and how to get the information.
- If the cluster is empty and you are able to run shorter jobs to evade the limits.

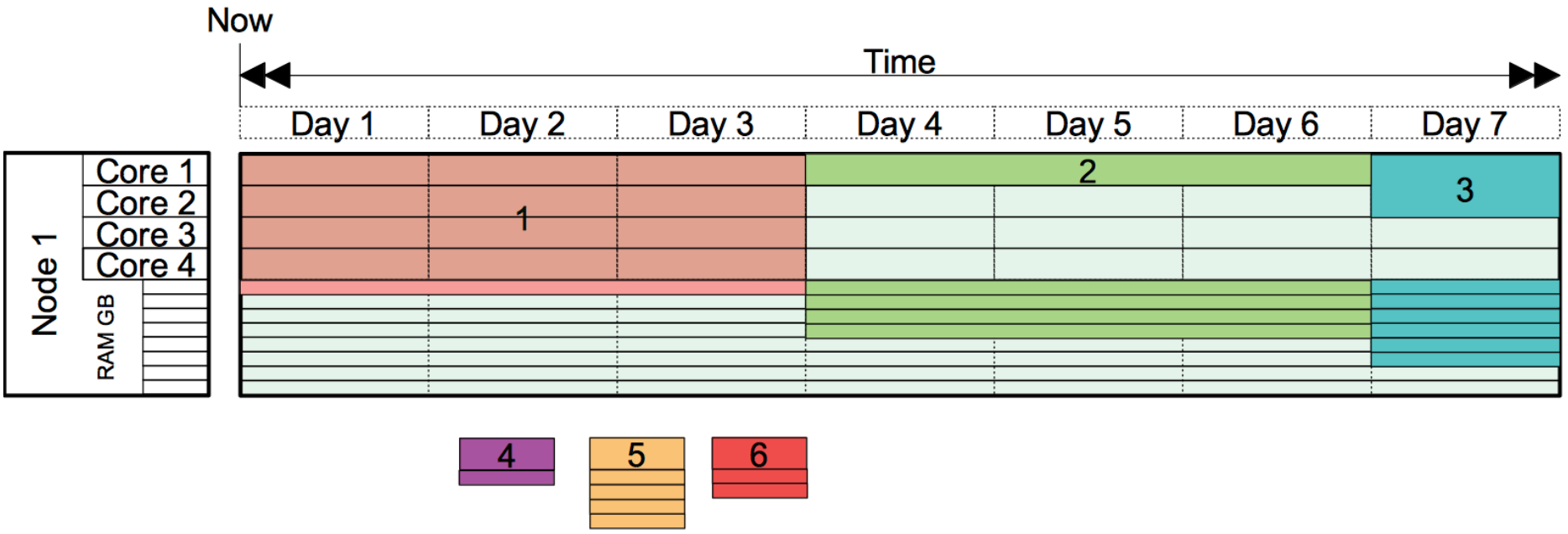
Scheduling Cores and Memory



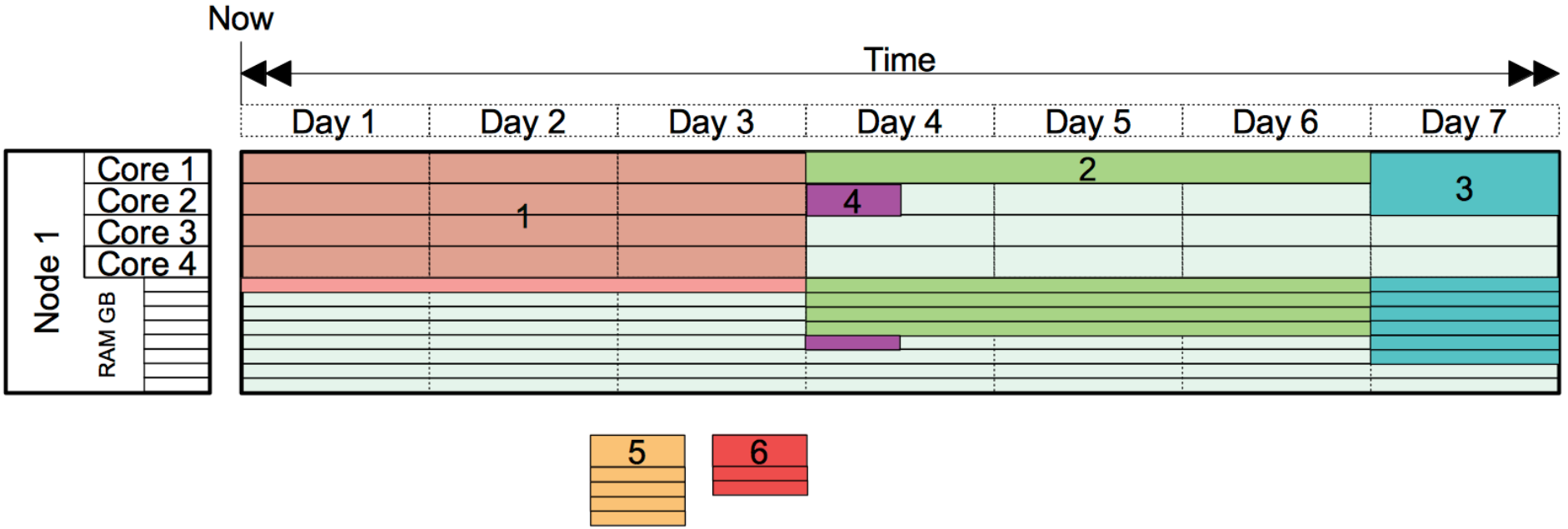
Scheduling Cores and Memory



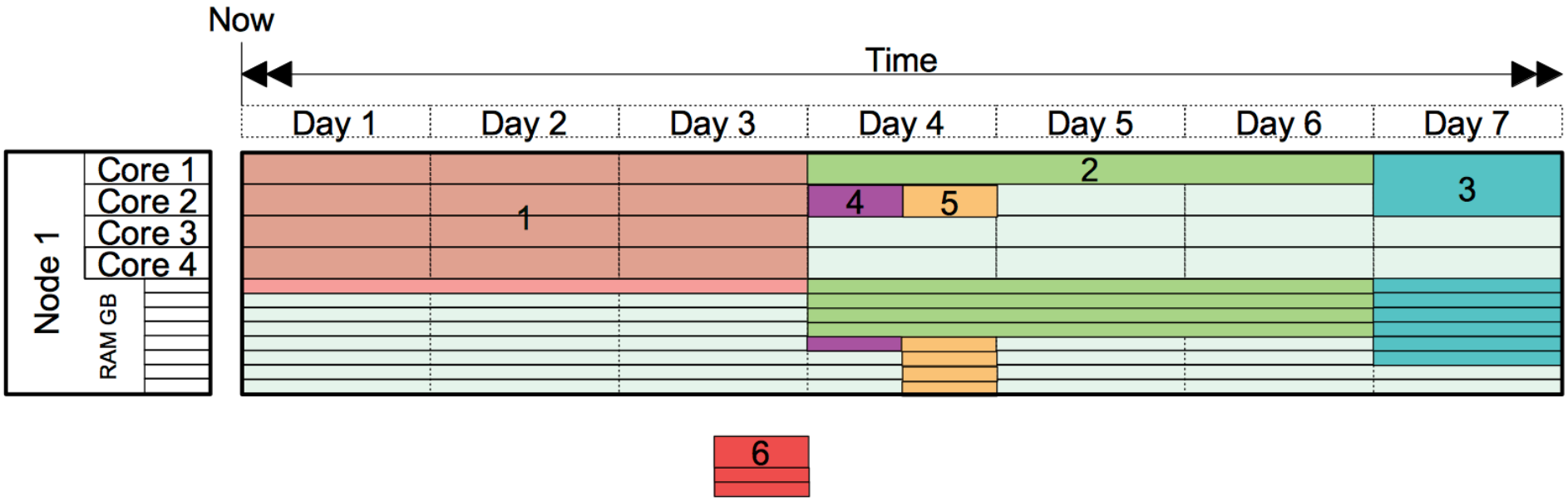
Scheduling Cores and Memory



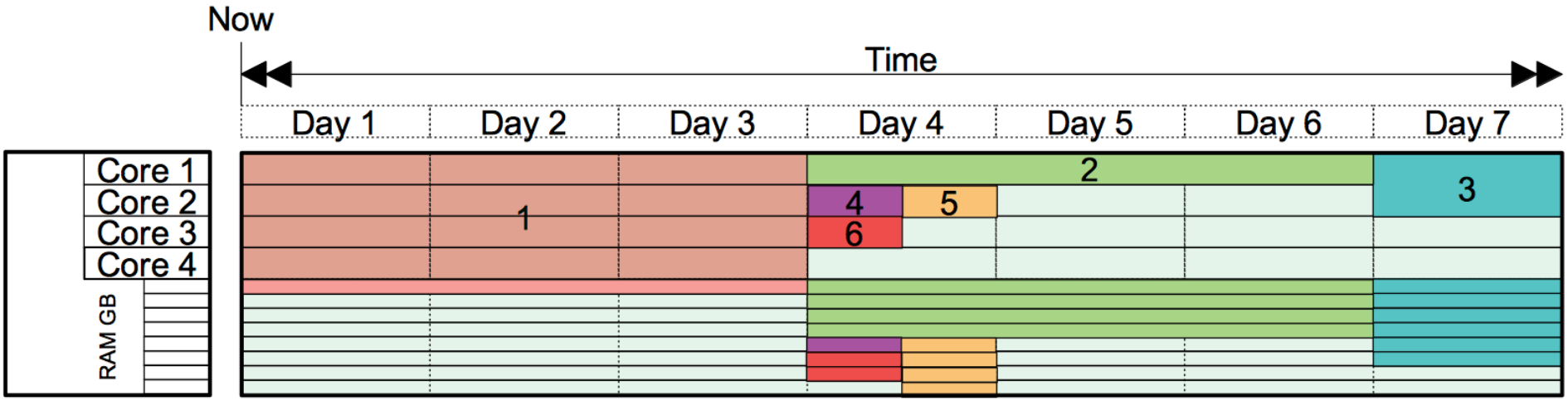
Scheduling Cores and Memory



Scheduling Cores and Memory



Scheduling Cores and Memory



Schedulable Resources

- Cores,
- Memory,
- GPUS,
- Disk,
- Software licenses

Basic Job submission

Job submission

What type of information does a scheduler needed to know schedule a job on a multiuser (shared) batch scheduled cluster?

Submitting a Job

- If you have a program that you wish to run you need to figure out the resource requirements of your Job. These requirements include:
  - walltime: maximum length of time your will take to run
  - number of cpus, memory, nodes, gpus
- The command to submit your job is sbatch, although sbatch allows you to specify your requirements on the command line, however you should put your requirements in a job script.
- sbatch jobscript.sh

Running out of runtime

If your jobs are running out of time.

- Ask for more time.
- Don’t ask for to much runtime.
- Asking for more runtime may limit you to how many resources can run your job. This may interact with how much memory you asked for.

Emailed about your job

Requires that you give the scheduler your email address and tell the scheduler when you wish to get notified.

The options are when your job:

begins, ends, fails, is requeued, or all the above.

#SBATCH --mail-user=no.email@ubc.ca

#SBATCH --mail-type=ALL

Job name and output

We can give a job a name see that instead of or in addition to the job number.

#SBATCH --job-name=my-named-job

We can also put the output of job into a specific file

%j is the jobs id number,

%x is the jobs name

By default the the output is: "slurm-%j.out”

#SBATCH –o my-output-file-%j.out

Simple slurm job script

```
#!/bin/bash
#SBATCH --ntasks=1
#SBATCH --nodes=1
#SBATCH --time=0-00:02
#SBATCH --mail-type=ALL
#SBATCH --mail-user=no.email@ubc.ca
#SBATCH –o my-output-file-%j.out
#SBATCH --job-name=my-named-job
sleep 1000;  # Replace with a line running code
```

Basic Slurm script commands

Slurm script command	Description
#!/bin/bash	Sets the shell that the job will be executed on the compute node
#SBATCH --ntasks=1 #SBATCH --n1	Requests for 1 processors on task, usually 1 cpu as 1 cpu per task is default.
#SBATCH --time=0-05:00 #SBATCH -t 0-05:00	Sets the maximum runtime of 5 hours for your job
#SBATCH --mail-user= <email>	Sets the email address for sending notifications about your job state.
#SBATCH --mail-type=BEGIN #SBATCH --mail-type=END #SBATCH --mail-type=FAIL #SBATCH --mail-type=REQUEUE #SBATCH --mail-type=ALL	Sets the scedualing system to send you email when the job enters the follwoing states: BEGIN,END,FAIL,REQUEUE,ALL
#SBATCH --job-name=my-named-job	Sets the Jobs name

Slurm Jobs and steps (Advanced topic)

- Slurm jobs have something called steps
- Each of these steps is like a job and may have different resources used in it.
- Steps are “mini-jobs” within a job.
- You can use the command **srun** to carry out each step
- Srun has a similar syntax to sbatch
- Prologue and epilogue scripts that run just before and after your job runs in a different step

Interactive Jobs

- One can ask for an interactive Job to run a program on the cluster and interact with it while it is running.
- Interactive jobs are useful for debugging.
- We can request an allocation of resources with the salloc command:

```
salloc --ntasks=1 --nodes=1 --time=0-01:20
```

SLURM Environment Variables

Environment Variable	Description
SLURM_JOB_NAME	User specified job name
SLURM_JOB_ID	Unique slurm job id
SLURM_NNODES	Number of nodes allocated to the job
SLURM_NTASKS	Number of tasks allocated to the job
SLURM_ARRAY_TASK_ID	Array index for this job
SLURM_ARRAY_TASK_MAX	Total number of array indexes for this job
SLURM_MEM_PER_CPU	Memory allocated per CPU
SLURM_JOB_NODELIST	List of nodes on which resources are allocated to Job
SLURM_JOB_CPUS_PER_NODE	Number of CPUs allocated per Node
SLURM_JOB_PARTITION	List of Partition(s) that the job is in.
SLURM_JOB_ACCOUNT	Account under which this job is run.

Running basic Jobs

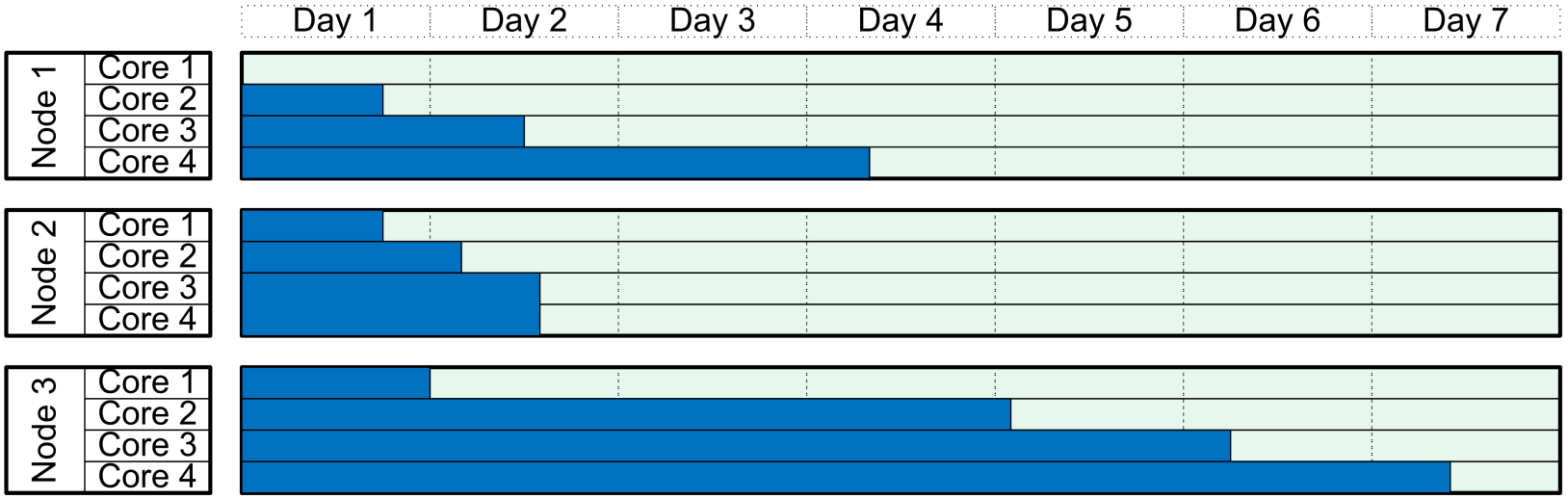
BREAK FOR PRACTICE

Jobs Types: Parallelism

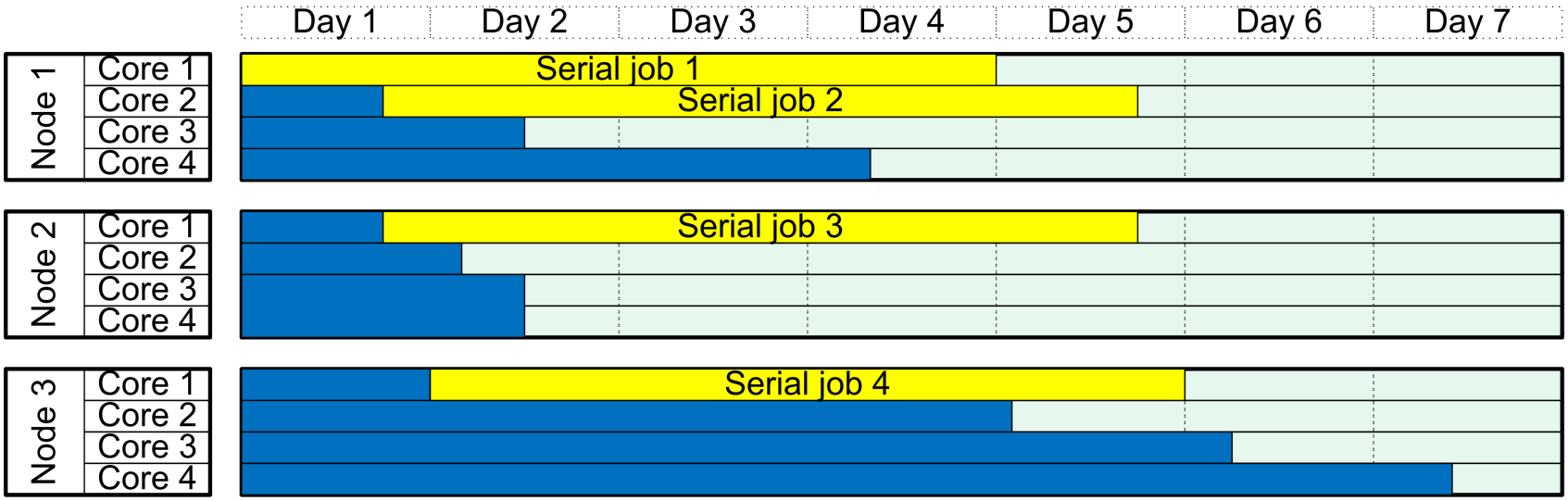
- Many Serial Jobs
- Message Passing (MPI)
- Single node muti-core (OpenMP, Gaussian)
- Hybrid/ Advanced

	1 Nodes	N Nodes
1 cpu	Serial	MPI
X cpus	OpenMP	Hybrid

Visualizing Multinode cluster



Many Serial Jobs



Many Serial Jobs

- Use 1 cpu per job
- Easiest and most efficient to schedule
- Excellent scaling linear speedup
- Example job would be a parameter searches
- In your slurm file one can ask for a serial job with:
- #SBATCH --ntasks=1

Slurm Serial Job Example

```
#!/bin/bash
#SBATCH --ntasks=1
#SBATCH --time=0-00:02
#SBATCH --mail-type=ALL
#SBATCH --mail-user=no.email@ubc.ca
#SBATCH –o my-output-file-%j.out
#SBATCH --job-name=my-named-job
sleep 1000;  # Replace with a line running code
```

Tips for running more Serial Jobs

- Submit shorter serial jobs
- Many short serial jobs will run before larger job
- Checkpoint longer jobs and submit them as short jobs, this will also save you when the cluster suffers hardware or power failure.

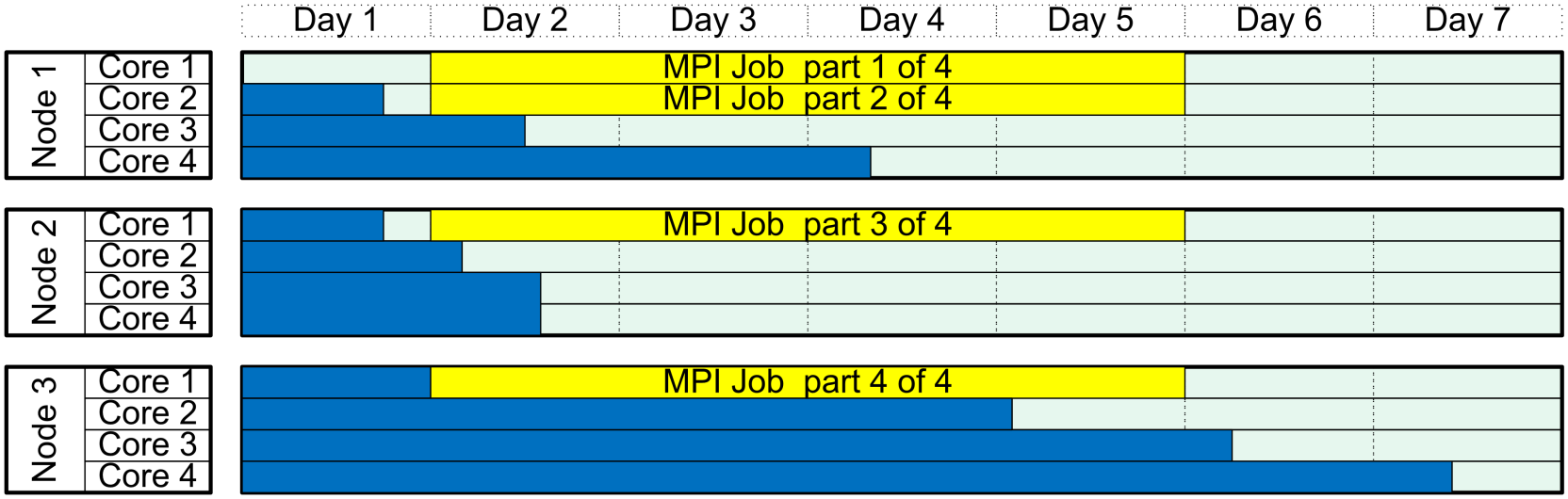
Job array

- Job arrays are used when you have need to submit a large number of Jobs using the same job script.
- There is a naming convention for jobs in array, which is useful as you don't need to remember a large number of unique job ids or job names: jobname[0]
- Job arrays are preferred as they don't require as much computation by the scheduling system to schedule, as they are evaluated as a group instead of individually. Ask for a job array in one of the following ways:
  - **#SBATCH --array=0-99**
    - job array 100 jobs numbered 0 -99
  - **#SBATCH --array=1,2,3,5,7**
    - Job array with 5 jobs with indexes [1,2,3,5,7]
  - **#SBATCH --array=0-99%5**
    - job array 100 jobs numbered 0 -99 with a maximum of 5 running at any time

Job array sample script

```
#!/bin/bash
#SBATCH --ntasks=1                      # Number of cores/tasks
#SBATCH --time=0-00:02                  # Runtime in D-HH:MM
#SBATCH --job-name=my-array-job         # Sets the Jobs name
#SBATCH --array=1-12                    # Ask for an Job array of
12 tasks
echo "This jobs name is:    $SLURM_JOB_NAME"
echo "This jobs jobid is:    $SLURM_JOB_ID”
echo "This jobs taskid is:  $SLURM_ARRAY_TASK_ID”
sleep 30
hostname
```

MPI job


Day 1
Day 2
Day 3
Day 4
Day 5
Day 6
Day 7
Node 1
Core 1
Core 2
Core 3
Core 4
MPI Job part 1 of 4
MPI Job part 2 of 4
Node 2
Core 1
Core 2
Core 3
Core 4
MPI Job part 3 of 4
Node 3
Core 1
Core 2
Core 3
Core 4
MPI Job part 4 of 4

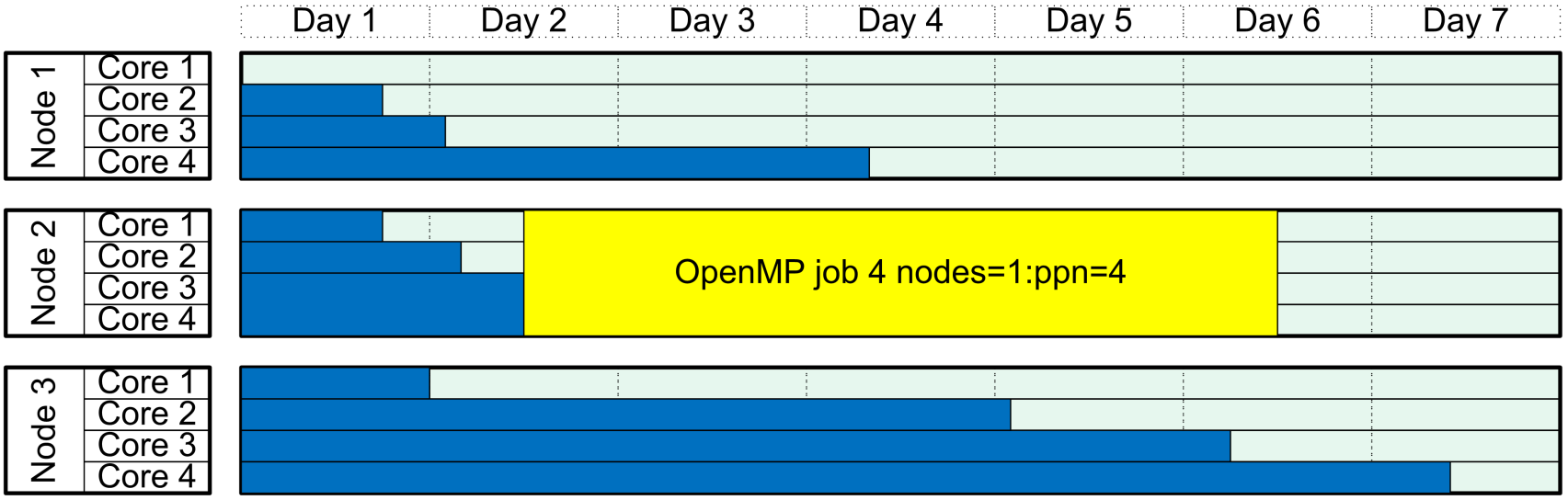
MPI Jobs

- Use the network for message passing
- Each job uses multiple cpus each of which can be on a different node.
- Each process uses a different memory address space
- More difficult to write parallel code than OpenMP as deadlocks are more common.
- Can scale higher than OpenMP as clusters are typically larger than even large SMP machines

MPI Job Submission

- This type of job can have its processes running on any node, multiple processes can run on a single node.
- #SBATCH --ntasks=X

Single node muti-core job (OpenMP, Gaussian, Threads)



Single node muti-core job

- All the threads must run on a single node.
- The threads share a single memory address space
- Can compile serial and parallel executables from the same source code
- OpenMP is one of the easiest methods of parallel programing, can be done incrementally.

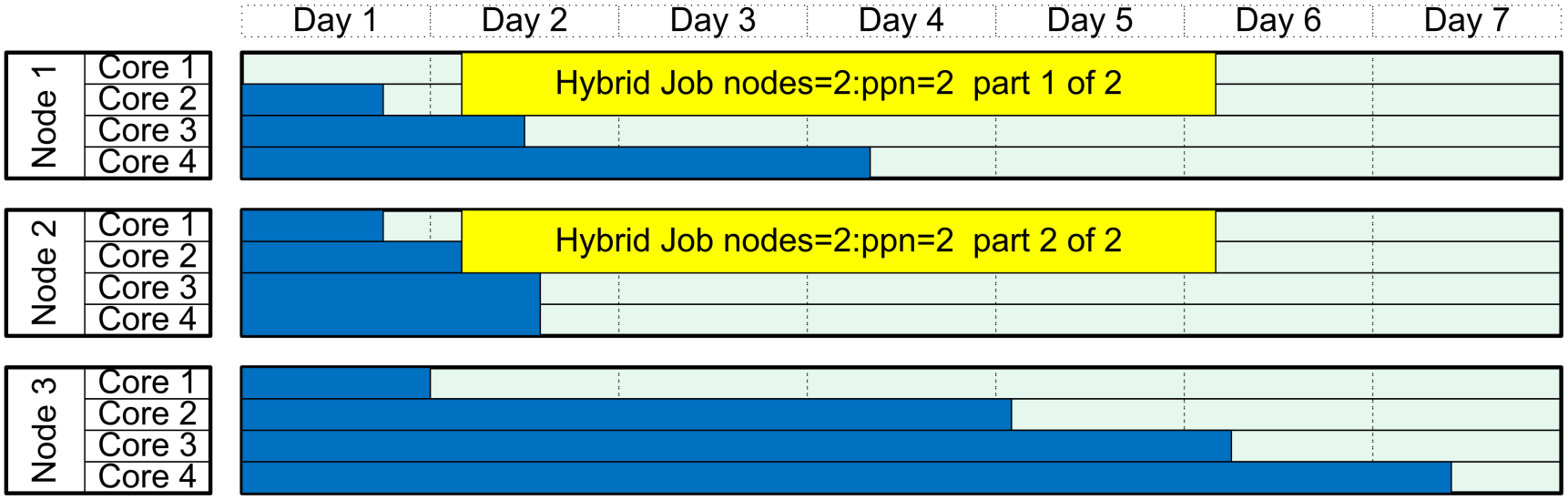
OpenMP job submission

- This type of job must have its thread running on one node, sharing the same memory.
- Communication between parts of the job is done via memory
- #SBATCH --cpus-per-task=X
- One can ask the program to run a number of threads via an environment variable:
  - export OMP_NUM_THREADS=8
- Usually set it to the requested cores:
  - export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

Tips for running OpenMP Jobs

- Check the state of the cluster to see if your job will run quickly.
- If you have a number of OpenMP style jobs you should consider running longer jobs using less cpus per job instead.
  - It is faster and more efficient to schedule single/ smaller processor jobs.
  - This advice may not apply when you need other resources like large amount of RAM per job.

Hybrid Job


Day 1
Day 2
Day 3
Day 4
Day 5
Day 6
Day 7
Node 1
Core 1
Core 2
Core 3
Core 4
Hybrid Job nodes=2:ppn=2 part 1 of 2
Node 2
Core 1
Core 2
Core 3
Core 4
Hybrid Job nodes=2:ppn=2 part 2 of 2
Node 3
Core 1
Core 2
Core 3
Core 4

Why use a hybrid job

- It's possible to combine OpenMP and MPI for running on clusters of SMP machines
- Need more memory or other resource than is available per core.
- Advanced systems of running parallel jobs can utilize resources more efficiently. Communication between cores is faster than between distant nodes. These systems include Chapel language as well as Partitioned global address space languages (PGAS) such as Unified Parallel C, Co-array Fortran.

Slurm script commands

Slurm script command	Description
#SBATCH −ntasks=X	Requests for X tasks. When cpus-per-task=1 (and this is the default) this requests X cores. When not otherwise constraint these CPUs may be running on any node
#SBATCH --nodes=X	Request that a minimum of X nodes be allocated to this job
#SBATCH --nodes=X-Y	Request that a minimum of X nodes and a maximum of Y nodes be allocated to this job
#SBATCH --cpus-per-task=X	Request that a minimum of X CPUs per task be allocated to this job
#SBATCH --tasks-per-node=X	Requests minimum of X task be allocated per node

Slurm script commands

Slurm script commands	Description of effects
#SBATCH --ntasks=1 #SBATCH --cpus-per-task=1	Requests 1 CPU (Serial) cpus-per-task is set to 1 by default and may be omitted.
#SBATCH --cpus-per-task=X #SBATCH --ntasks=1 #SBATCH --nodes=1	Requests for X CPUs in 1 task on 1 node (OpenMP) Both ntasks and nodes are set to 1 by default and may be omitted
#SBATCH --ntasks=X #SBATCH --tasks-per-node=X #SBATCH --cpus-per-task=1	Requests for X CPUs and tasks on 1 node (OpenMP) cpus-per-task is set to 1 by default and may be omitted.
#SBATCH --ntasks=X #SBATCH --nodes=1 #SBATCH --cpus-per-task=1	Requests for X CPUs and tasks on 1 node (OpenMP) cpus-per-task is set to 1 by default and may be omitted.

Slurm script commands

Slurm script commands	Description of effects
#SBATCH --ntasks=X #SBATCH --cpus-per-task=1	Requests X CPUs and tasks (MPI) cpus-per-task is set to 1 by default and may be omitted.
#SBATCH --ntasks=X #SBATCH --ntasks-per-node=Y #SBATCH --cpus-per-task=1	Requests for X CPUs and tasks with Y CPUs and tasks per node (MPI) cpus-per-task is set to 1 by default and may be omitted.
#SBATCH --ntasks=X #SBATCH --nodes=1 #SBATCH --cpus-per-task=1	Requests for X CPUs and tasks on the same node, cpus-per-task is set to 1 by default and may be omitted.
#SBATCH --ntasks=X #SBATCH --nodes=1 #SBATCH --cpus-per-task=1	Requests for X CPUs and tasks on the 1 node cpus-per-task is set to 1 by default and may be omitted.

Serial, mpi, openmp, hybrid, jobarrays

BREAK FOR PRACTICE

Slurm Jobs and memory

- Always ask for slightly less than total memory on node as some memory is used for OS, and your job will not start until enough memory is available.
- You may specify the maximum memory available to your job in one of 2 ways.
  - Ask for a total memory used by your jobs (MB)

    #SBATCH --mem=4000
  - Ask for memory used per process/core in your job (MB)

    #SBATCH --mem-per-cpu=2000

Slurm Jobs and memory

It is very important to specify memory correctly

- If you don’t ask for enough and your job uses more ,your job will be killed.
- If you ask for too much, it will take a much longer time to schedule a job, and you will be wasting resources.
- If you ask for more memory than is available on the cluster your job will never run. The scheduling system will not stop you from submitting such a job or even warn you.
- If you don’t specify any memory then your job will get a very small default maximum memory.

Find out how much memory a job used.

Command	Flags	What its used for
sstat		Display various status information of a running job
	−j <jobid>	Displays information about the specified job
	--format= AveCPU,MaxRSS,MaxVMSize,JobID	limits the information to that about memory (MaxVMSize is requested memory) (MaxRSS is memory used)
sacct		Displays slurm accounting data
	−j <jobid>	Displays information about the specified job
	-u $USER	Displays information about jobs belong to a specific user
	--format= JobID,AveCPU,MaxRSS,MaxVMSize	limits the information to that about memory
salloc		Submit to run Job Interactively, use unix system utilities such as top
	Same flags as sbatch	Note not all sbatch flags work

Virtual and Physical memory

A program can ask an OS (operating system) to use a chunk of memory. This is virtual or requested memory: MaxVMSize

MaxRSS is the amount of memory used by your code.

These can and often are different because the OS grants memory request without actually giving the memory until it us used.

Interactive Jobs for debugging

Use salloc instead of sbatch to launch interactive jobs.

```
salloc  --ntasks=4 --mem-per-cpu 4000 -t 0-00:20
```

List environment variables with printenv

```
printenv | grep ^S
```

To check memory usage use

```
top –u $USER
```

Remember that you are logged in only on one machine and your job may span more than one

Slurm jobs and GPUS

- To request GPU use the following syntax
  - **#SBATCH --gres=gpu:1**
- Slurm recognize GPUs and knows the state of the GPU.
- Different sites have different GPUs some have more than one type of GPU.
- Too request a large gpu node on cedar
  - **#SBATCH --gres=gpu:lgpu:4**

Slurm script commands

PBS script command	Description
#SBATCH --mem=4000	Requests 4000 MB of memory in total
#SBATCH --mem-per-cpu=4000	Requests 4000 MB of memory per cpu
#SBATCH --licenses=sas:2	Requests 2 SAS licenses
#SBATCH --gres=gpu:1	Requests that your job get 1 GPU allocated per node
#SBATCH --exclusive	Requests that your job run only on nodes with no other running jobs
#SBATCH --dependency=after:job_id1	Requests that the the job start after job (jobid1) has **started**
#SBATCH --dependency=afterany:job_id1, job_i2	Requests that the the job start after ether job (jobid1) or job (jobud2) has **finished**
#SBATCH --dependency=afterok:job_id1	Requests that the the job start after job (jobid1) has **finished successfully**

Memory, Features, Software licenses , Partitions

BREAK FOR PRACTICE

Job Submission Requiring Exclusive Access

- Sometimes there is a need for exclusive access to guarantee that no other job will be running on the same nodes as your job such as during debugging.
- To guarantee that the job will only run on nodes without other jobs you own use:
  - **#SBATCH --exclusive**
- Your research group may get charged for using the whole node and not just the resources requested, and it may take a long time to gather resources needed for these special jobs.

Job submission multiple projects

- If you are part of two different Compute Canada projects and are running jobs for both, you need to specify the accounting group for each project so that the correct priority of the job can be determined and so that the usage is “charged” to the correct group.
- In order to specify an accounting group for a Job use:
  - **#SBATCH --account=accouting_group**
- You can see your accounting group information with the “sacctmgr show user <username> withassoc” command.

Job dependencies

- If you want one job to start one after another finishes use the
  - **#SBATCH --dependency=afterok:job_id1**
- If one can break apart a long job into several shorter jobs then the shorter jobs will often be able to be ran faster. This is also the technique to use if the required job runtime is longer than the maximum walltime allowed on the cluster.
  - **job1id=$(sbatch anwser-q24.1.sh| awk '{print $4}')**
  - **sbatch --dependency=aftercorr:$job1id anwser-q24.2.sh**

Temporary available local storage

- Some software like Gaussian needs to make many small reads and writes to disk. The cluster (lustre) file system cannot do this well and this becomes a performance problem for the job and the cluster its running on.

- Each node has local disk, that is shared by all jobs running on the node. One specifies the requests the local storage via “#PBS –l file=1000mb”.

- There is a directory created for each job when it is run. When the job finished this directory is automatically erased. The directory name is $TMPDIR. A example of using the temporary local storage:
  - **#SBATCH --tmp=200G**

    cd $SLURM_TMPDIR
    <run my job >
    mkdir my_new_dir
    cp <file I wish to save> my_new_dir/

Node types on Cedar

Number of Nodes	% of total	Memory per core (GiB)	Total Mem (GiB)	Cores	GPUS	Partition type
640	54.2%	4	192	48		cpubase
576	32.5%	4	128	32		
182	10.3%	8	256	32		
24	1.4%	16	512	32		cpularge
24	1.4%	48	1536	32		
4	0.2%	96	3072	32		
114	46.3%	4 (32 per GPU)	128	24	4	gpubase
132	53.7%	8 (64 per GPU)	256	24	4	gpularge

Node types on Graham

Number of Nodes	% of total	Memory per core (GiB)	Total Mem (GiB)	Cores	GPUS	Partition type
884	91.42 %	4	128	32		cpubase
56	5.79 %	8	256	32		
24	2.48 %	16	512	32		cpularge
3	0.31 %	96	3072	32		
160	100.00%	4 (64 per GPU)	128	32	2	gpubase

Node types on Beluga

Number of Nodes	% of total	Memory per core (GiB)	Total Mem (GiB)	Cores	GPUS	Partition type
172	24.16%	2.4	96	40		cpubase
516	72.47%	4.8	192	40		
24	3.37%	19.2	768	40		cpularge
172	100.00%	4.8 (48 per GPU)	192	40	4	gpubase

Partitions, or Queues

You can the have more than one queue or partition on a cluster

The partitions can be for the same or different sets of resources

Unlike queues in real life that are first in first out (FIFO) on a clusters we allocate by priority.

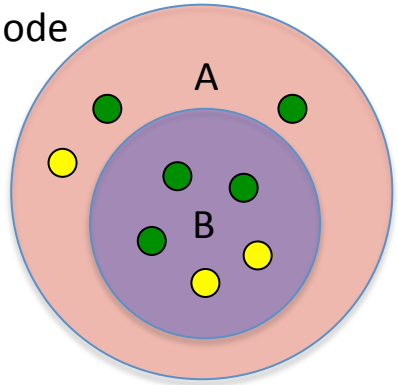
Partitions

- Your job will automatically be assigned
- A node can be in multiple partitions simultaneously
- A job can be in multiple partitions simultaneously, and can have multiple per-partition priorities.

Partitions, Nodes and Jobs

Idle node

Busy node



Partition B has 5 nodes 3 are idle

Partition A has 8 nodes 5 are idle

On the 3 nodes in partition A but not in partition can only be assigned jobs from partition A

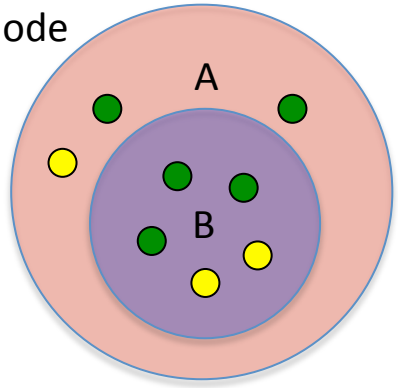
On the 5 nodes in partition B (that are also in partition A) can be assigned jobs from Partitions A or B

The highest priority job in partition A if it uses more than 2 nodes may not scheduled if there is a higher priority job in partition B.

Partitions, Nodes and Jobs

Idle node

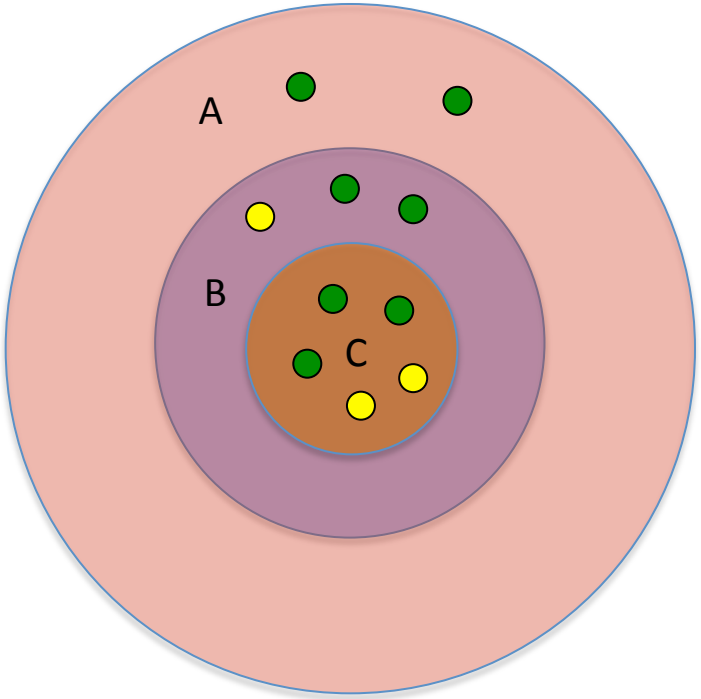
Busy node



If jobs in partition A have a higher priority than the jobs in partition B all the idle nodes including those in partition B will be assigned to jobs from partition A

If Jobs in partition B have higher priority than jobs in partition B they will run in partition B. The highest priority jobs in partition A that can fit into the 2 idle nodes will be started on the 2 nodes that are not in partition B.

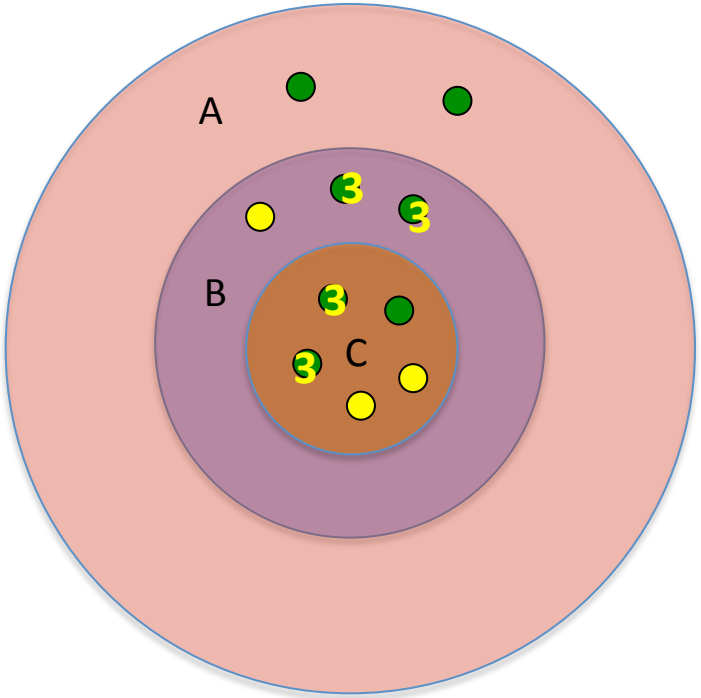
Partitions and priority Exercise 1



- Idle node
- Busy node

Job ID	Partition	Priority	Resources
1	C	High	2 nodes
2	A	Medium	1 node
3	B	Highest	4 nodes
4	A	low	1 node

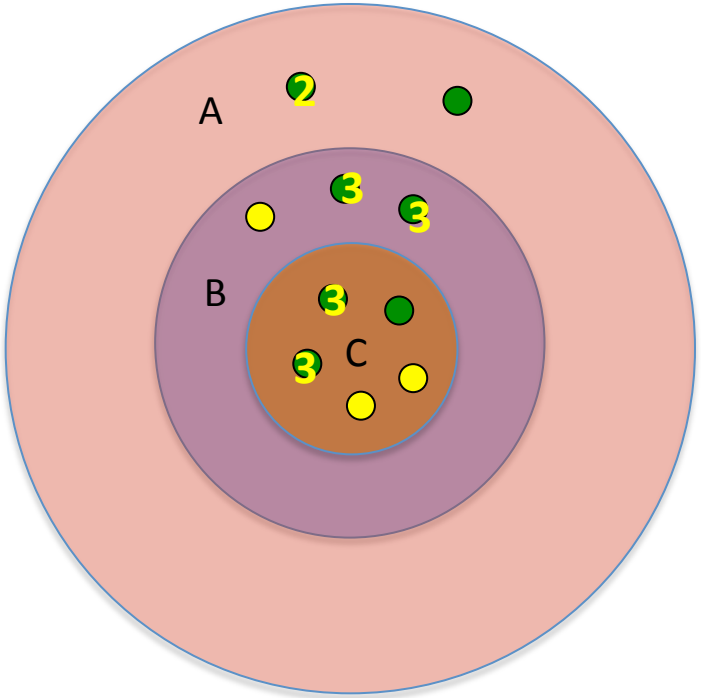
Partitions and priority Exercise 1



- Idle node
- Busy node

Job ID	Partition	Priority	Resources
1	C	High	2 nodes
2	A	Medium	1 node
3	B	Highest	4 nodes
4	A	low	1 node

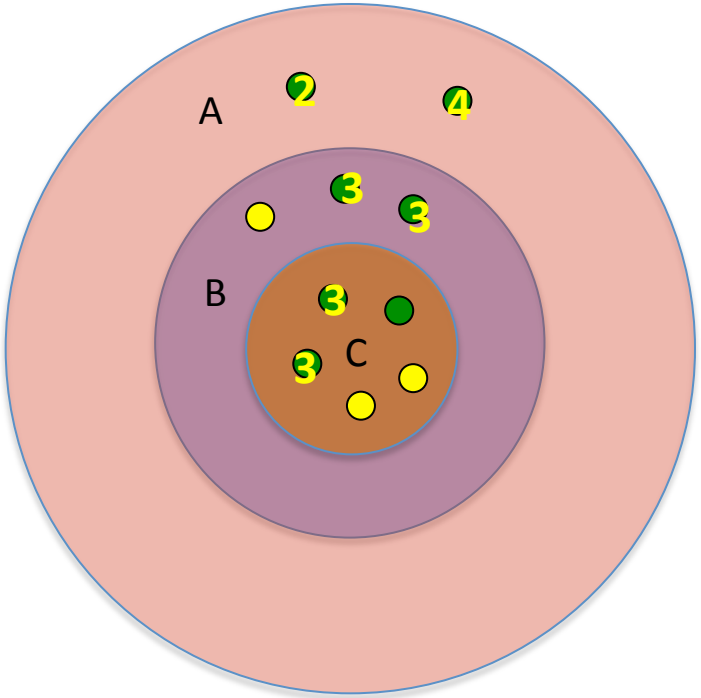
Partitions and priority Exercise 1



- Idle node
- Busy node

Job ID	Partition	Priority	Resources
1	C	High	2 nodes
2	A	Medium	1 node
3	B	Highest	4 nodes
4	A	low	1 node

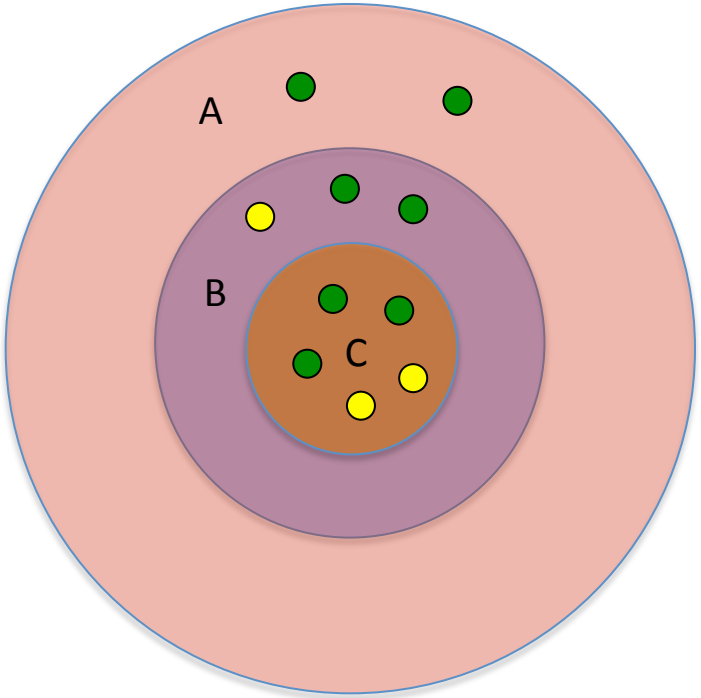
Partitions and priority Exercise 1



- Idle node
- Busy node

Job ID	Partition	Priority	Resources
1	C	High	2 nodes
2	A	Medium	1 node
3	B	Highest	4 nodes
4	A	low	1 node

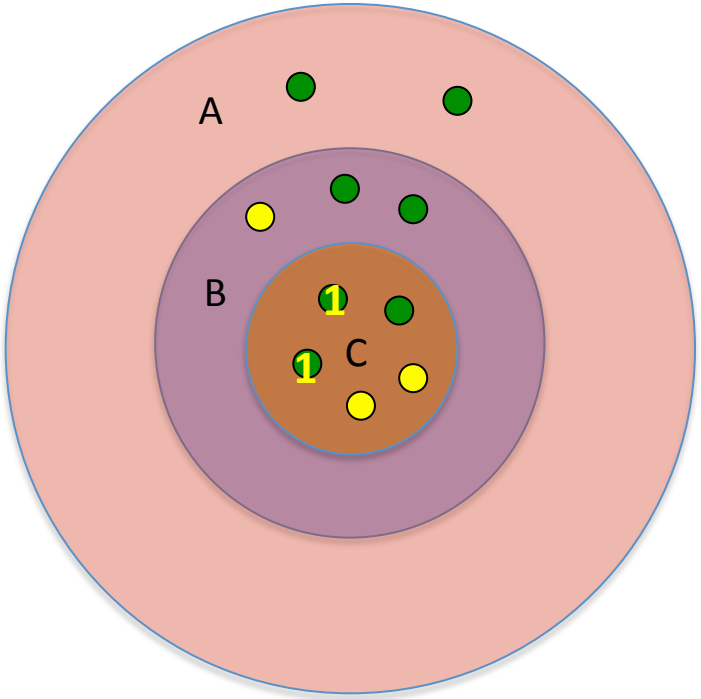
Partitions and priority Exercise 2



- Idle node
- Busy node

Job ID	Partition	Priority	Resources
1	C	Highest	2 nodes
2	A	High	4 node
3	B	low	2 nodes
4	A	lowest	1 node

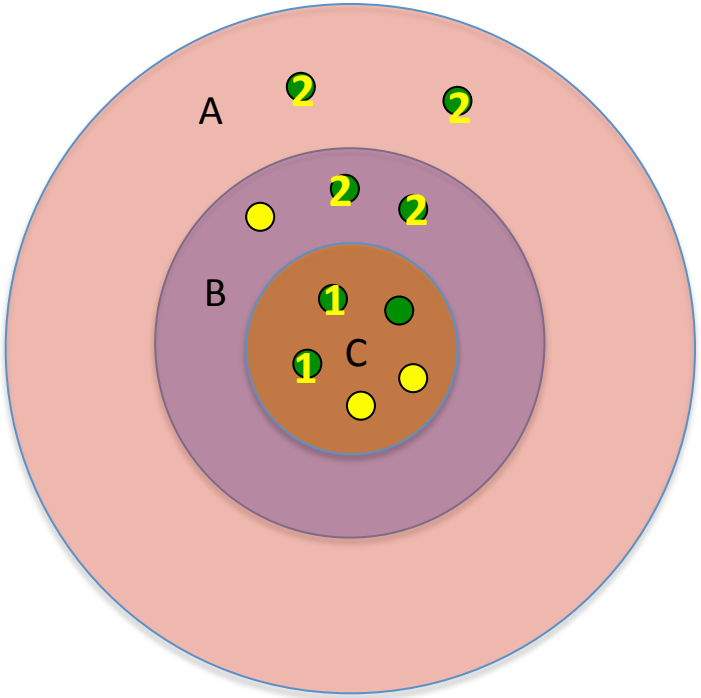
Partitions and priority Exercise 2



- Idle node
- Busy node

Job ID	Partition	Priority	Resources
1	C	Highest	2 nodes
2	A	High	4 node
3	B	low	2 nodes
4	A	lowest	1 node

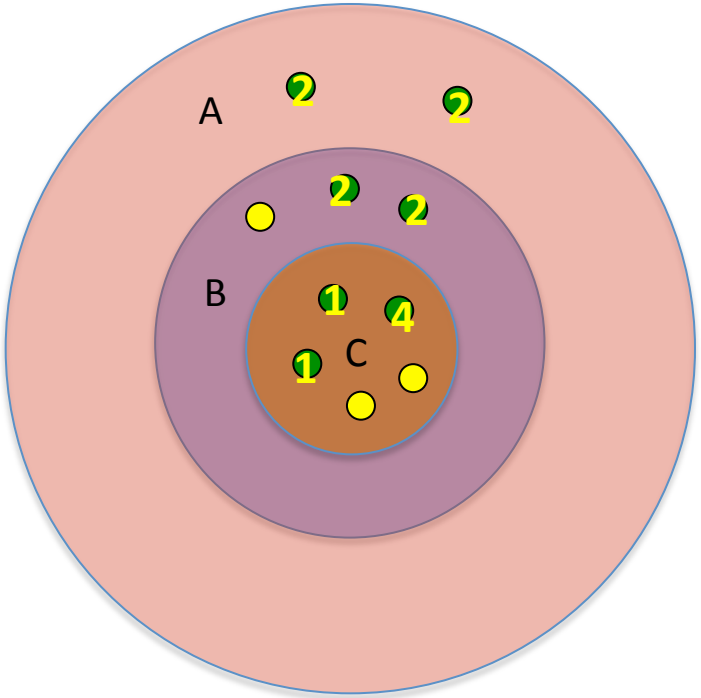
Partitions and priority Exercise 2



- Idle node
- Busy node

Job ID	Partition	Priority	Resources
1	C	Highest	2 nodes
2	A	High	4 node
3	B	low	2 nodes
4	A	lowest	1 node

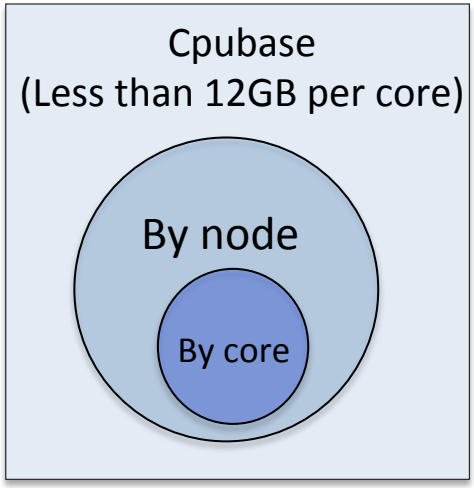
Partitions and priority Exercise 2

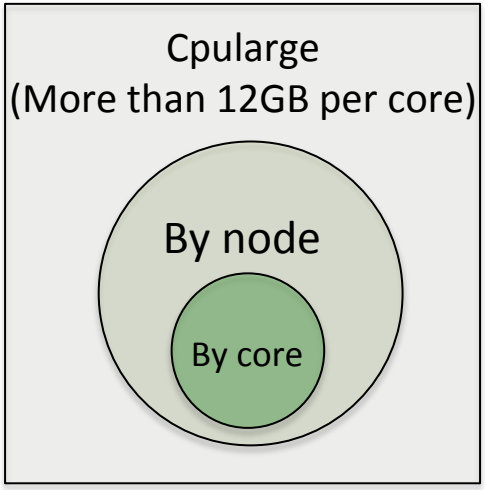


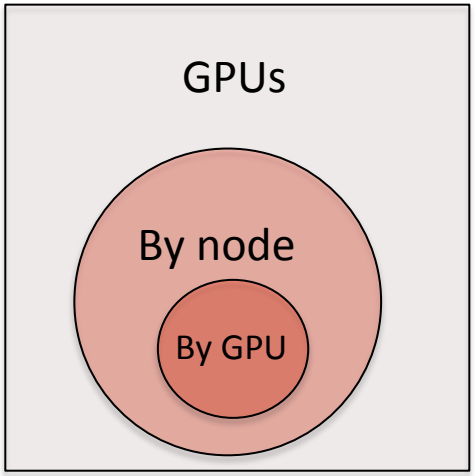
- Idle node
- Busy node

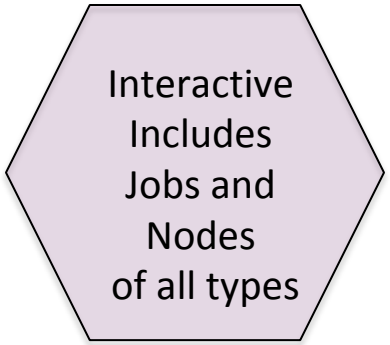
Job ID	Partition	Priority	Resources
1	C	Highest	2 nodes
2	A	High	4 node
3	B	low	2 nodes
4	A	lowest	1 node

Partitions on Cedar, Graham and Beluga









Why do we have by-node?

1. Responsiveness in order to minimize turnaround time and
2. Fairness in order to allocate fairly to whole node users

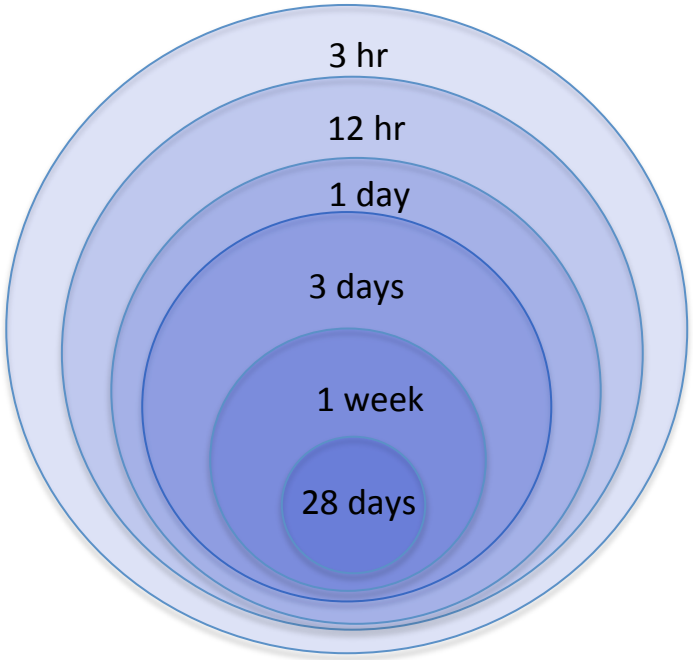
If we allowed serial jobs to run on all nodes, the chances that there was a node that had all 32 cores would be very small.

If ½ the cluster was empty and the job distributed randomly the chances a any particular node to be empty = $\frac{1}{2^{32}} = \frac{1}{4,294,967,296}$

Why do we have by-core?

1. Efficiency, not all jobs need 32 core, 128GiB of RAM etc…
2. Fairness it would not fair to charge you for 32 if you need one and had no way of asking for less than 32,

Partitions and Walltime in Compute Canada



Partition name	Maximum walltime
*_b1	3 hours
*_b2	12 hours
*_b3	1 day
*_b4	3 days
*_b5	7 days
*_b6	28 days

The shorter walltime partitions include all the nodes of longer walltime partitions.

The walltimes of partitions are maximums. Your job ends up in the shortest walltime partition that it can fit in.

Why do we allow long walltimes?

Fairness, some researcher's jobs need to run a long time

There are many **disadvantages**

- Jobs with a long walltime result in a **reduction** of **fairness**, **efficiency**, **responsiveness** of the cluster.
- Clusters suffer outages, longer jobs are more affected.
- If we allow long walltimes on most resources, this can create a perverse incentive a vicious cycle were in order to maximize runtime after waiting so long for other jobs to finish longer jobs are submitted.

$$\frac{1}{2^{32}} = \frac{1}{4,294,967,296}$$

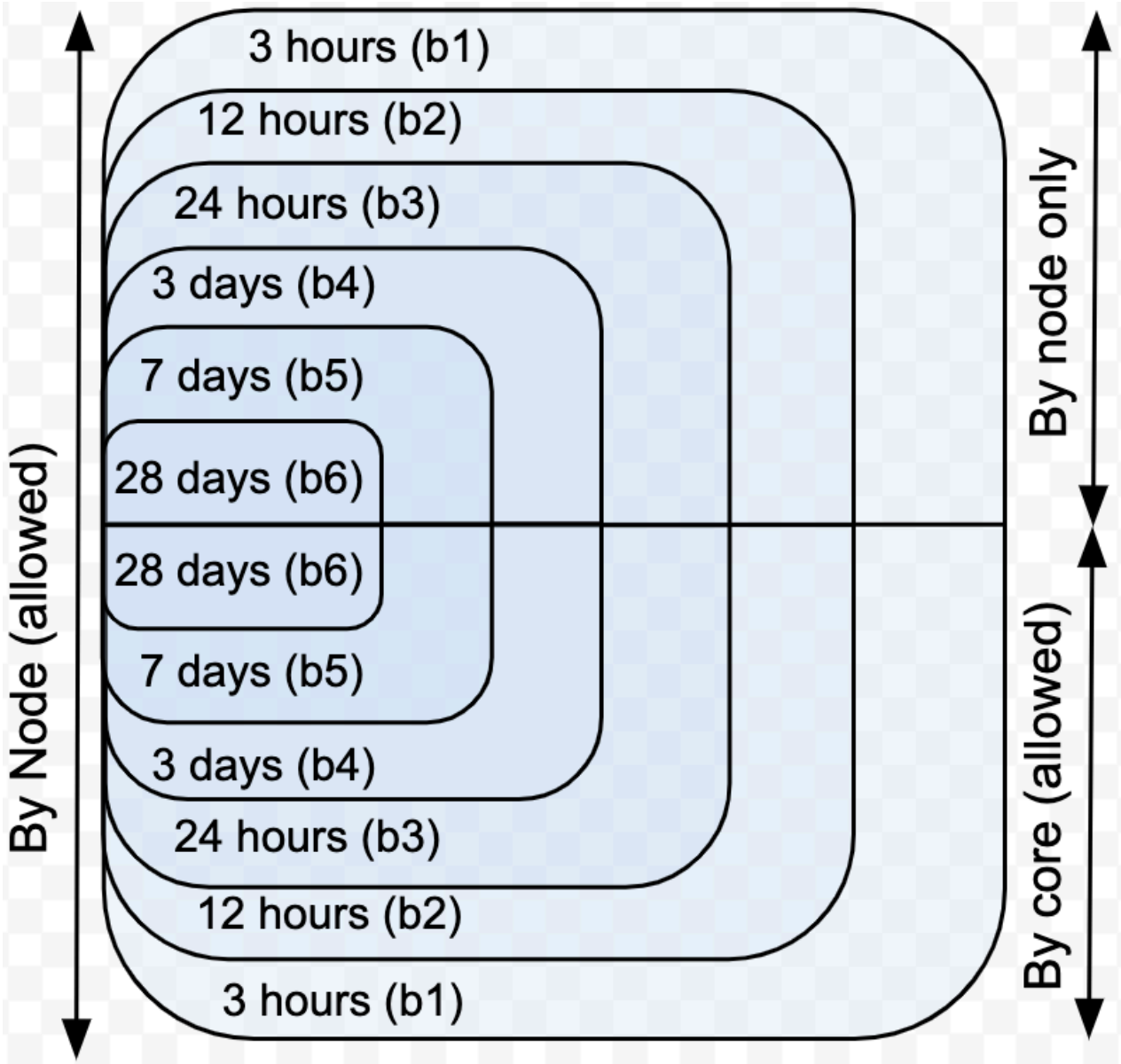
Why do we set dedicated short walltime partitions and nodes?

Fairness, some researchers have a iterative workflow they need to run short jobs and analyze the result before running another. A system with long walltimes is unusable to them.

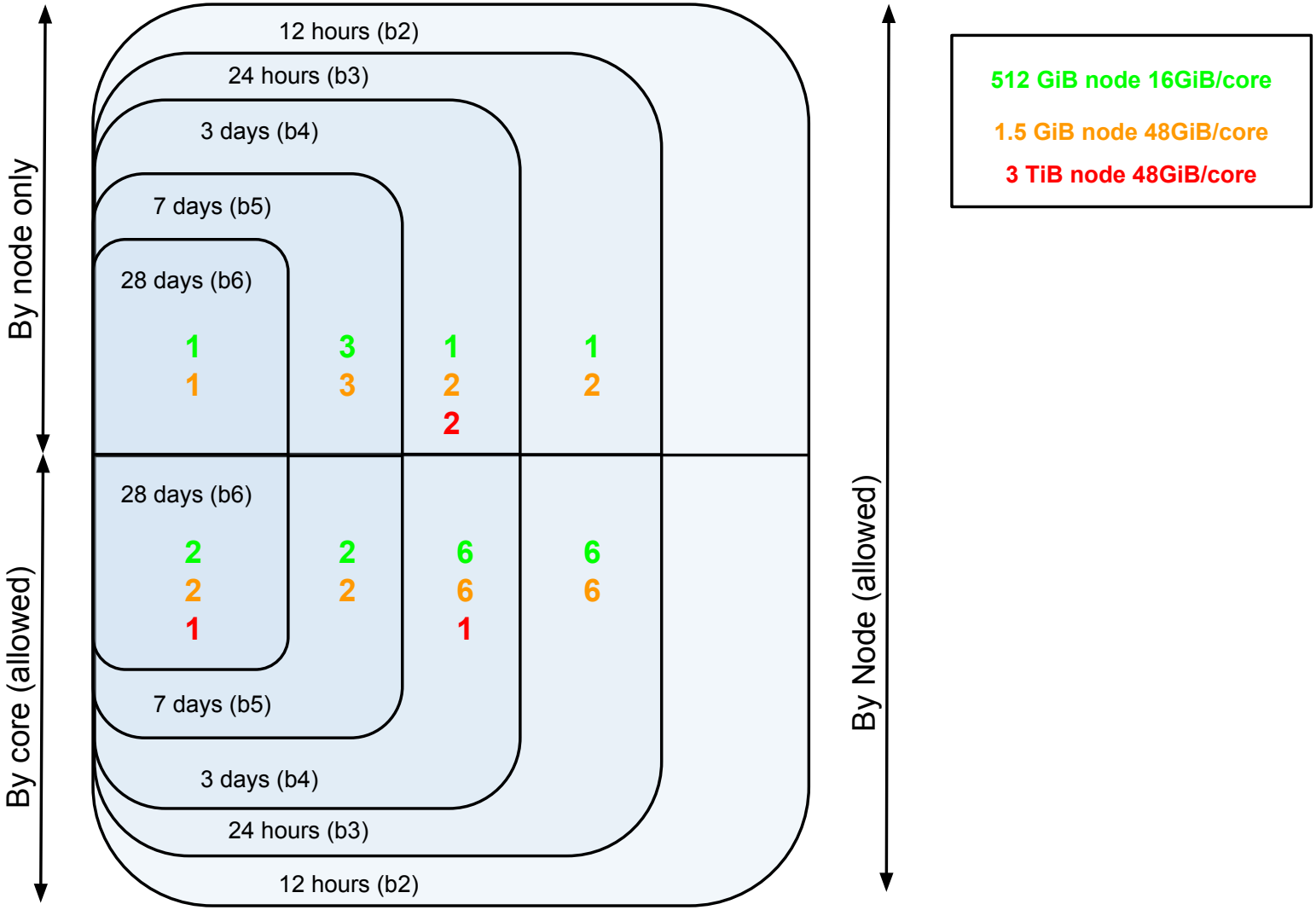
There are many **advantages** to shorter jobs

- Clusters with a short walltime are **fairer**, **efficient**, and **responsive**.
- Little work is lost in Clusters outages if jobs are shorter
- Breaking up a longer job into shorter sections is called **checkpointing** and restart.

Partitions in CC



The large memory partitions on Cedar by node type and max wall time



Partition Stats

(CC script)

Node type	Max walltime					
	3 hr	12 hr	24 hr	72 hr	168 hr	672 hr
Number of Queued Jobs by partition Type (by node:by core)						
Regular	1:15	2:31	2:145	11:187	86:69	3:2
Large Mem	0:0	0:0	0:0	0:0	0:1	0:1
GPU	0:1	0:526	10:10	0:0	189:4	0:0
Number of Running Jobs by partition Type (by node:by core)						
Regular	60:6	4:2	45:836	5:90	11:1065	1:4
Large Mem	0:0	0:0	0:0	0:0	0:0	1:0
GPU	0:20	2:10	13:2	0:0	0:0	0:3
Number of Idle nodes by partition Type (by node:by core)						
Regular	0:0	0:0	0:0	0:0	0:0	0:0
Large Mem	3:1	0:0	0:0	0:0	0:0	0:0
GPU	17:1	11:1	0:0	0:0	0:0	0:0
Total Number of nodes by partition Type (by node:by core)						
Regular	851:411	821:391	756:346	636:276	180:100	90:50
Large Mem	27:12	24:11	24:11	20:3	3:2	2:1
GPU	156:78	144:72	116:58	104:52	13:12	13:12

Partition and node visualization

We have a tool in development that shows you how many nodes your job can run on.

https://slurm-wizard.uofa.c3.ca

Partition Exercise 1

Your group member is trying to run the following job on Graham has submitted it and is waiting for 3 weeks. What is the problem, and what should be done about it.

```
#!/bin/bash
#SBATCH --mail-user=nouser@ubc.ca
#SBATCH --mail-type=ALL
#SBATCH --nodes=1
#SBATCH --gres=gpu:4
#SBATCH --ntasks=32
#SBATCH --mem-per-cpu=4000
#SBATCH --time=24:00
```

Partition Exercise 2

Your group member is trying to run the following job on Graham and has submitted it and is waiting for 2 weeks. What is the problem, and what should be done about it

```
#!/bin/bash
#SBATCH --mail-user=nouser@ubc.ca
#SBATCH --mail-type=END
#SBATCH --time=03:00
#SBATCH --ntasks=48
#SBATCH --mem-per-cpu=1000
```

Partition Exercise 3

Your group member is trying to run the following job on Cedar and has submitted it and is waiting for 3 weeks. What is the problem, and what should be done about it

```
#!/bin/bash
#SBATCH --mail-user=nouser@ubc.ca
#SBATCH --mail-type=ALL
#SBATCH --mem-per-cpu=12500
#SBATCH --time=7-12:00
#SBATCH --nodes=1
#SBATCH --ntasks=32
```

Partition Exercise 4

Your group member is trying to run the following job on Cedar and has submitted it and is waiting for 3 weeks. What is the problem, and what should be done about it

```
#!/bin/bash
#SBATCH --mail-user=nouser@ubc.ca
#SBATCH --mail-type=ALL
#SBATCH --mem=1600000
#SBATCH --time=3-06:00
#SBATCH --nodes=1
#SBATCH --ntasks=32
```

Slurm script commands

PBS script command	Description
#SBATCH --mem=4000	Requests 4000 MB of memory in total
#SBATCH --mem-per-cpu=4000	Requests 4000 MB of memory per cpu
#SBATCH --licenses=sas:2	Requests 2 SAS licenses
#SBATCH --gres=gpu:1	Requests that your job get 1 GPU allocated per node
#SBATCH --exclusive	Requests that your job run only on nodes with no other running jobs
#SBATCH --dependency=after:job_id1	Requests that the the job start after job (jobid1) has **started**
#SBATCH --dependency=afterany:job_id1, job_i2	Requests that the the job start after ether job (jobid1) or job (jobud2) has **finished**
#SBATCH --dependency=afterok:job_id1	Requests that the the job start after job (jobid1) has **finished successfully**

Slurm script commands

PBS script command	Description
#SBATCH --account=acc_name	To submit a job to a specific accounting group such as RAC/RAS allocation or different role
#SBATCH --tmp=200G	Asks for 200Gb of temporary disk space
#SBATCH --constraint=blue	To ask for a node feature or constraint set by cluster admin. Here we are looking for “blue” nodes.
#SBATCH --partition=partition_name	To ask for the job to run in a specific partition or queue by name, (unlike Moab there can be more than 1 partition per Job)
--prolog=<executable>	Run by srun only, runs the executable before the step
--epilog=<executable>	Run by srun only, runs the executable after the step finishes

SLURM Environment Variables

Environment Variable	Description
SLURM_JOB_NAME	User specified job name
SLURM_JOB_ID	Unique slurm job id
SLURM_NNODES	Number of nodes allocated to the job
SLURM_NTASKS	Number of tasks allocated to the job
SLURM_ARRAY_TASK_ID	Array index for this job
SLURM_ARRAY_TASK_MAX	Total number of array indexes for this job
SLURM_MEM_PER_CPU	Memory allocated per CPU
SLURM_JOB_NODELIST	List of nodes on which resources are allocated to Job
SLURM_JOB_CPUS_PER_NODE	Number of CPUs allocated per Node
SLURM_JOB_PARTITION	List of Partition(s) that the job is in.
SLURM_JOB_ACCOUNT	Account under which this job is run.

Job submission practice

BREAK FOR PRACTICE

Getting information on your Job

Command	What its used for
squeue -u <username>	List all current jobs for a user
squeue -u <username> -t PENDING	List all pending jobs for a user
squeue -u <username> -t RUNNING	List all running jobs for a user
Squeue -p <partitionname>	List all the jobs in a partition
scontrol show job <jobid>	List information on Job
scontrol show jobid -dd <jobid>	List detailed information on Job
Squeue -o "%.18i %.30P %.8j %.8u %.2t %.8p %.10M %.6D %R "	Formatted output of squeue: we added priority and made the partition field bigger (30 characters)

Getting information on your Job

Command	What its used for
sstat --format=AveCPU,MaxRSS,MaxVMSize,JobID -j <jobid>	List info resource used by your completed job : average cpu time, Max memory, Max virtual memory, JobId
sacct −u <username> --format=JobID,JobName,AveCPU,MaxRSS,MaxVMSize,JobID,Elapsed	List resources used by all jobs of a user
sprio	List job priority information

squeue

```
kamil@zeno ~]$ squeue

JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
2020_1 mem12_sho my-array    kamil  R       0:04      1 zeno001
2020_4 mem12_sho my-array    kamil  R       0:04      1 zeno001
2019 mem12_sho my-named    judy  R       0:11      1 zeno001
```

Squeue command for user
Squeue -u $USER

```
[kamil@zeno ~]$ squeue -u kamil

JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
2025 mem12_sho anwser-q    kamil  R       0:01      1 zeno001
597520 cpubase_b aln_ERR1    kamil PD       0:00      1 (Dependency)
597540 cpubase_b aln_SRR9    kamil PD       0:00      1 (Dependency
598316 cpubase_b aln_SRR2    kamil PD       0:00      1 (DependencyNeverSa
598324 cpubase_b aln_SRR2    kamil PD       0:00      1 (DependencyNeverSa
```

Squeue command for queued jobs
squeue -u <username> -t PENDING

```
[kamil@zeno ~]$ squeue -u kamil -t pending
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
597520 cpubase_b aln_ERR1    kamil PD       0:00      1 (Dependency)
597540 cpubase_b aln_SRR9    kamil PD       0:00      1 (Dependency
598316 cpubase_b aln_SRR2    kamil PD       0:00      1 (DependencyNeverS
598324 cpubase_b aln_SRR2    kamil PD       0:00      1 (DependencyNeverS
619783 cpubase_b ala1805S    kamil PD       0:00      1 (Priority)
617318 cpubase_b Pseudomo    kamil PD       0:00      1 (Resources)
617319 cpubase_b Pseudomo    kamil PD       0:00      1 (Resources)
```

squeue -u <username> -t RUNNING

```
[kamil@cedar ~]$ squeue -u kamil -t running
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
2026 mem12_sho   anwser-q    kamil  R       0:02      1 zeno001
620930 cpubase_b HRAGR001    kamil  R      23:58      1 cdr57
617805 cpubase_b     Ro:0    kamil  R    9:44:23      4 cdr[72,88,92,95]
584942 cpubase_b runmpi.s    kamil  R 2-11:09:29      4 cdr[81-83,98]
574866 cpubase_b Ro:-0.08    kamil  R 2-22:21:17      5 cdr[77,79-80,84,9
618505 cpubase_b Bowtie2_    kamil  R    9:42:10      1 cdr215
```

Jobs by partition

squeue -p <partitionname>

```
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(R
535639 cpubase_b AE17631.    kamil PD       0:00      1 (Resource
591830 cpubase_b    bz.sh   erming PD       0:00      1 (Resource
615762 cpubase_b AE21380.    kamil PD       0:00      1 (Resource
401219 cpubase_b CTD095.s     john PD       0:00      1 (Resource
491576 cpubase_b gen3x1s8     judy  R 2-08:04:59      1 cdr747
535638 cpubase_b AE17594.    kamil  R 1-11:46:03      1 cdr101
491574 cpubase_b gen3x1s6    masao  R 4-20:06:44      1 cdr79
491575 cpubase_b gen3x1s7    masao  R 4-20:06:44      1 cdr85
```

Squeue queries

Squeue -o "%.18i %.30P %.8j %.8u %.2t %.8p %.10M %.6D %R " -u <username>

```
[kamil@cedar5 test]$ squeue -o "%.18i %.30P %.8j %.8u %.2t %.8p %.10M %.6D %R " -u kamil
JOBID                      PARTITION     NAME     USER ST PRIORITY       TIME  NODES NODELIST(REASON)
597520 cpubase_bycore_b1,cpubackfill aln_ERR1    kamil PD 0.001164       0:00      1 (Depend
597540 cpubase_bycore_b1,cpubackfill aln_SRR9    kamil PD 0.001164       0:00      1 (Depend
597592 cpubase_bycore_b1,cpubackfill aln_SRR5    kamil PD 0.001164       0:00      1 (Depend
597593 cpubase_bycore_b1,cpubackfill aln_SRR8    kamil PD 0.001164       0:00      1 (Depend
```

scontrol show job <jobid>

```
[kamil@zeno ~]$ scontrol show job  2026
JobId=2026 JobName=anwser-q3.sh
   UserId=kamil(1005) GroupId=slurmteam(1007) MCS_label=N/A
   Priority=38885 Nice=0 Account=team1 QOS=mem12_short
   JobState=COMPLETED Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:00:31 TimeLimit=00:02:00 TimeMin=N/A
   SubmitTime=2017-03-22T13:51:02 EligibleTime=2017-03-22T13:51:02
   StartTime=2017-03-22T13:51:02 EndTime=2017-03-22T13:51:33 Deadline=N/A
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   Partition=mem12_short AllocNode:Sid=zeno:31494
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=zeno001
   BatchHost=zeno001
   NumNodes=1 NumCPUs=1 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=1,mem=1948M,node=1
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryCPU=1948M MinTmpDiskNode=0
   Features=(null) Gres=(null) Reservation=(null)
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/home/kamil/anwser-q3.sh
   WorkDir=/home/kamil
   StdErr=/home/kamil/slurm-q1-2026.err
   StdIn=/dev/null
```

Priority
sprio

JOBID	PRIORITY	AGE	FAIRSHARE	PARTITION	TRES
130976	7088	2500	0	625	cpu=2526,mem=1437
167003	6150	2500	0	1250	cpu=2008,mem=392
195802	4996086	2500	4991771	833	cpu=469,mem=45,gres/
195809	4996086	2500	4991771	833	cpu=469,mem=45,gres/
195810	4996086	2500	4991771	833	cpu=469,mem=45,gres/
205281	8206	2500	0	625	cpu=1875,mem=1800,gr
205290	6408	2500	0	625	cpu=1875,mem=2,gres/
544814	23534	1741	21571	208	cpu=13,mem=2
544815	23534	1741	21571	208	cpu=13,mem=2
617580	24194	373	22768	1042	cpu=10,mem=2
617581	24194	373	22768	1042	cpu=10,mem=2

Demonstration on cluster

- SSH cluster and show all the following commands and how to interpret them
  - squeue
  - squeue -u $USER
  - squeue -t pending
  - squeue -t running
  - squeue -p <partition>
  - squeue (custom format)
  - scontol show job <jobid>
  - Sprio -n

Job information practice

BREAK FOR PRACTICE

QUESTIONS?

Scheduling and Job Management 3

Using a cluster effectively

Presentation contents

Priority, Allocations and Fairshare

Cluster limits, Reservations and Topology

Getting information on your Cluster

Trouble shooting your jobs

Priority

- Can only be positive in slurm.
- Only relative priority matters.
- Jobs with highest or least negative priority get reservation to run first.
- Highest priority job may not run first.

  A job which is using a small amount of resources that are in great supply may easily run before a high priority job requesting scarce or already used resources.
- In Compute Canada priority is determined per group via “fairshare” and how long your job sits in the queue
- “sprio” will show priority of your job

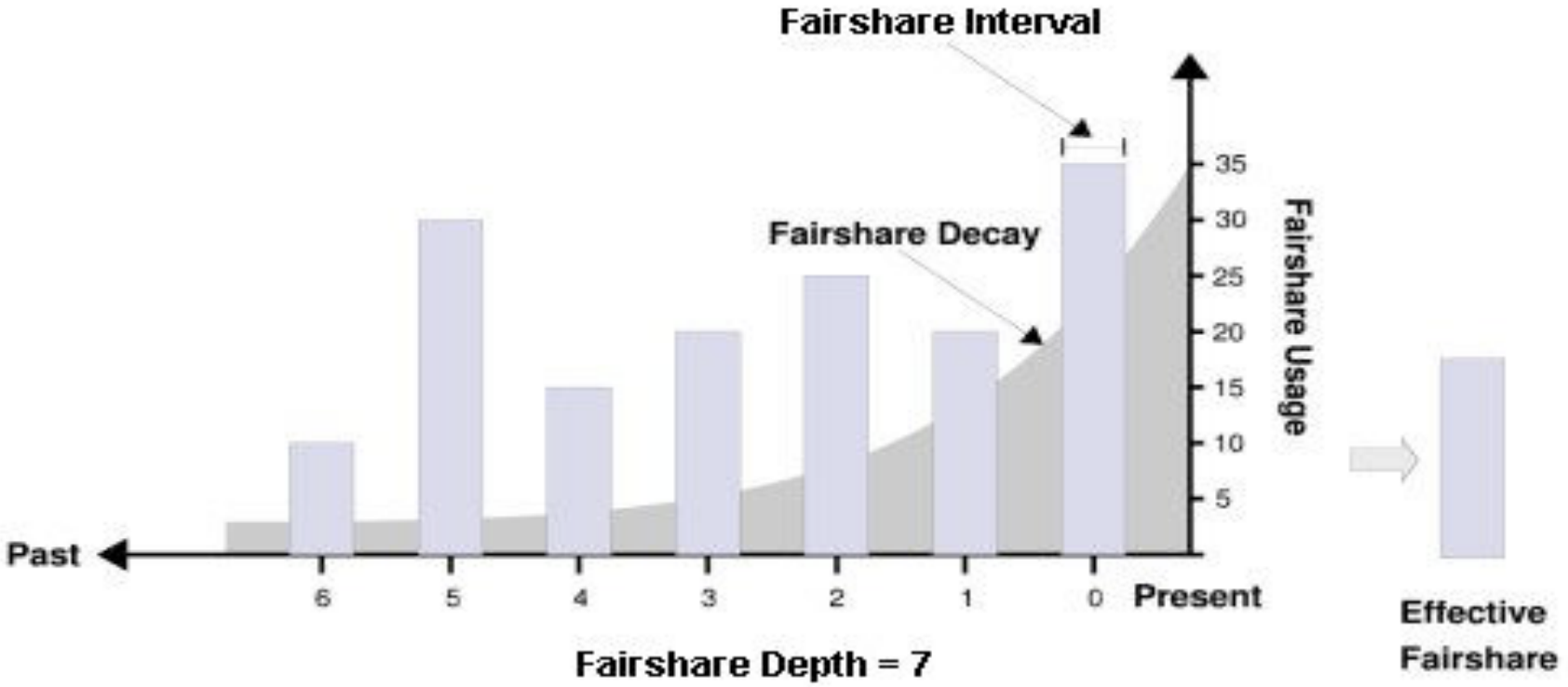
Priority
sprio

JOBID	PRIORITY	AGE	FAIRSHARE	PARTITION	TRES
130976	7088	2500	0	625	cpu=2526,mem=1437
167003	6150	2500	0	1250	cpu=2008,mem=392
195802	4996086	2500	4991771	833	cpu=469,mem=45,gres/
195809	4996086	2500	4991771	833	cpu=469,mem=45,gres/
195810	4996086	2500	4991771	833	cpu=469,mem=45,gres/
205281	8206	2500	0	625	cpu=1875,mem=1800,gr
205290	6408	2500	0	625	cpu=1875,mem=2,gres/
544814	23534	1741	21571	208	cpu=13,mem=2
544815	23534	1741	21571	208	cpu=13,mem=2
617580	24194	373	22768	1042	cpu=10,mem=2
617581	24194	373	22768	1042	cpu=10,mem=2

Fairshare

- Fairshare is a mechanism that allows historical resource utilization information to be incorporated into job feasibility and priority decisions
- In SLURM priority ranges from 1 to 0
- In Compute Canada fairshare compares your group's target usage to your group's actual usage during a time period. If your group has used less than your group share you are given higher priority.

Fairshare



- Fair share usage is weighted by when the usage occurred recent usage is more important then usage at the end of the period

Fairshare CPUs and GPUs and Equivalents

- We use GPU or CPU equivalent resources in all our calculations.
  - We “charge” for the largest share of resources on the typical node that was requested.
  - If your job uses all (memory/disk/any other resource) on a node and half the CPUs the scheduling system will “charge” your group as if you used all the CPUs on that node.
- Separate accounting groups for CPUs and GPUs
- For GPU jobs we only count number of GPU used or GPU equivalent in terms of other resources.

Group's Status: "sshare "

```
[kamil@cedar5 workshop_test]$ sshare | egrep "(--|Account|^root|no_rac_|ras_b|cc-debug|kamil_)"
             Account       User  RawShares  NormShares    RawUsage   NormUsage  EffectvUsage  FairShare
-------------------- ---------- ---------- ----------- ----------- ----------- ------------- ----------
root                                          1.000000 56519806629365289                      1.000000   0.500000
 no_rac_cpu                          3083     0.123572 54311297258252622    0.960925      0.960925   0.004562
  ras_basic_cpu                      3083     0.123532 54311297258252622    0.960925      0.960925   0.004554
   cc-debug_cpu                         1     0.000031      120455    0.000000      0.000237   0.004554
    cc-debug_cpu          kamil          1     0.000000           0    0.000000      0.000001   0.004554
   def-kamil_cpu                        1     0.000031 46106596622    0.000001      0.000238   0.004470
    def-kamil_cpu         kamil          1     0.000031 46106596622    0.000001      0.000238   0.004470
 no_rac_gpu                           75     0.003006 842007112518017    0.014898      0.014898   0.032224
  ras_basic_gpu                       75     0.002967 842007112518017    0.014898      0.014898   0.030781
   cc-debug_gpu                         1     0.000001       37224    0.000000      0.000004   0.030781
    cc-debug_gpu          kamil          1     0.000000           0    0.000000      0.000000   0.030781
   def-kamil_gpu                        1     0.000001 37555979258    0.000001      0.000004   0.016416
    def-kamil_gpu         kamil          1     0.000001 37555979258    0.000001      0.000004   0.016416
```

Priority

- Priority from **fairshare** is the most **important** part of priority it is not the only component.
- In Compute Canada's case we give a small amount of extra priority for jobs that have been waiting in the queue for a long time.
  - Named: **age_factor**
  - Without an age factor a larger job by a user with a small allocation may never run.
- Compute Canada sometimes gives extra priority to jobs from certain partitions. You can usually ignore this.
  - Named: **partition_factor**
  - This may be used to give high priority on a set of resources to a group who purchases the nodes and contributes them to a cluster allowing other people to run short low priority jobs, when they are not using them.

Group's Status: "sshare -l"

```
[kamil@cedar5 workshop_test]$ sshare  -l | egrep "(--|Account|^root|no_rac_|ras_b|cc-debug|kamil_)"
             Account       User  RawShares  NormShares    RawUsage   NormUsage  EffectvUsage  FairShare LevelFS
-------------------- ---------- ---------- ----------- ----------- ----------- ------------- ---------- ----------
root                                          0.000000 639083114320110                      1.000000
 no_rac_cpu                           1320    0.043194 404703982221822    0.633257      0.633257              0.068209
  ras_basic_cpu                       1320    0.999243 404703982221822    0.633257      1.000000              0.999243
   cc-debug_cpu                          1    0.000236  1273287234    0.000002      0.000003             75.104409
    cc-debug_cpu          kamil          1    0.004386           0    0.000000      0.000000   0.026537        inf
   def-kamil_cpu                         1    0.000236           0    0.000000      0.000000                   inf
    def-kamil_cpu         kamil          1    1.000000           0    0.000000      0.000000   0.486678        inf
 no_rac_gpu                             65    0.002127 6883285083841    0.010771      0.010771              0.197479
  ras_basic_gpu                         65    0.984848 6883285083841    0.010771      1.000000              0.984848
   cc-debug_gpu                          1    0.000236       12668    0.000000      0.000000            128389.386733
    cc-debug_gpu          kamil          1    0.004386           0    0.000000      0.000000   0.508693        inf
   def-kamil_gpu                         1    0.000236           0    0.000000      0.000000                   inf
    def-kamil_gpu         kamil          1    1.000000           0    0.000000      0.000000   0.973463        inf
```

Priority
sprio –n

```
kamil@cedar5 test]$ sprio | head

[kamil@cedar5 workshop_test]$ sprio -n
JOBID PRIORITY   AGE        FAIRSHARE  PARTITION  TRES

130976 0.00000165 1.0000000  0.0000000  0.2500000  cpu=0.17,mem=0.10
167003 0.00000143 1.0000000  0.0000000  0.5000000  cpu=0.13,mem=0.03
195802 0.00116324 1.0000000  0.9983542  0.3333333  cpu=0.03,mem=0.00,gr
195804 0.00116324 1.0000000  0.9983542  0.3333333  cpu=0.03,mem=0.00,gr
195807 0.00116324 1.0000000  0.9983542  0.3333333  cpu=0.03,mem=0.00,gr
195809 0.00116324 1.0000000  0.9983542  0.3333333  cpu=0.03,mem=0.00,gr
195810 0.00116324 1.0000000  0.9983542  0.3333333  cpu=0.03,mem=0.00,gr
```

Priority
sprio

JOBID	PRIORITY	AGE	FAIRSHARE	PARTITION	TRES
130976	7088	2500	0	625	cpu=2526,mem=1437
167003	6150	2500	0	1250	cpu=2008,mem=392
195802	4996086	2500	4991771	833	cpu=469,mem=45,gres/
195809	4996086	2500	4991771	833	cpu=469,mem=45,gres/
195810	4996086	2500	4991771	833	cpu=469,mem=45,gres/
205281	8206	2500	0	625	cpu=1875,mem=1800,gr
205290	6408	2500	0	625	cpu=1875,mem=2,gres/
544814	23534	1741	21571	208	cpu=13,mem=2
544815	23534	1741	21571	208	cpu=13,mem=2
617580	24194	373	22768	1042	cpu=10,mem=2
617581	24194	373	22768	1042	cpu=10,mem=2

Multiple allocations/accounting groups

- Occurs when group gets a RAC (Resource Allocation Competition) allocation and therefore a new allocation that becomes the default allocation.
- Occurs when a user is part of multiple Compute Canada research groups. One can select the default allocation, even a default allocation per cluster and send an email to support@westgrid.ca.
- In order to specify a accounting group to charge and figure out the priority use the following example in your job submission script.
  - **#SBATCH --account=accounting_group_name**

Allocations

- What does an allocation usually mean?
  - If you request average resources continually through the time period and run jobs, you are guaranteed to get at least your allocated resources over the time period (year).
- What if I have not applied for an allocation?
  - you have a default allocation

Allocations

- It is impossible for an allocation to be defined as: “Any time you ask for the resources allocated you will receive them”.
  - If 2 users are given 50% of a cluster each, and both don’t start running jobs until the 6th month they both cannot get the same cluster
- Unless an extraordinary situation exist allocation will not mean that the specified resources are available sitting idle.
  - Funding agencies don’t like to see resources sitting idle
  - An example of a extraordinary situation would be an Tsunami warning center which may need to have an allocation sitting idle so that when a earthquake occurs they can compute which beaches get hit and concentrate first responder resources to save lives.

Allocations in Compute Canada

- Compute Canada (CC) Resource Allocation Competition (RAC) is a Committee of researchers that evaluate proposed allocations on the basis of scientific merits and resources available. There is also a preliminary technical evaluation which evaluates the application on technical merits, job requirements. The technical evaluation reports its findings and recommendations to the RAC.
- Allocations are done yearly, the RAC call for proposals goes out every October.
- For more information see: https://www.westgrid.ca/support/accounts/resource_allocations

Getting information on you and your group

Command	What its used for
sacctmgr list Users USERS=<username>	List user and their default account (accounting group)
sacctmgr show user <username> withassoc	List user and their default account (accounting group) and shows more extensive information
sshare	Shows usage info for user usage and priority.
sshare -l	Shows even more info for user usage and priority.

Priority for your job

Compare it to other job

Fairshare target allocation to your group

Your groups usage by members

BREAK FOR PRACTICE

Other reasons that effect jobs starting

Cluster Limits

Reservations

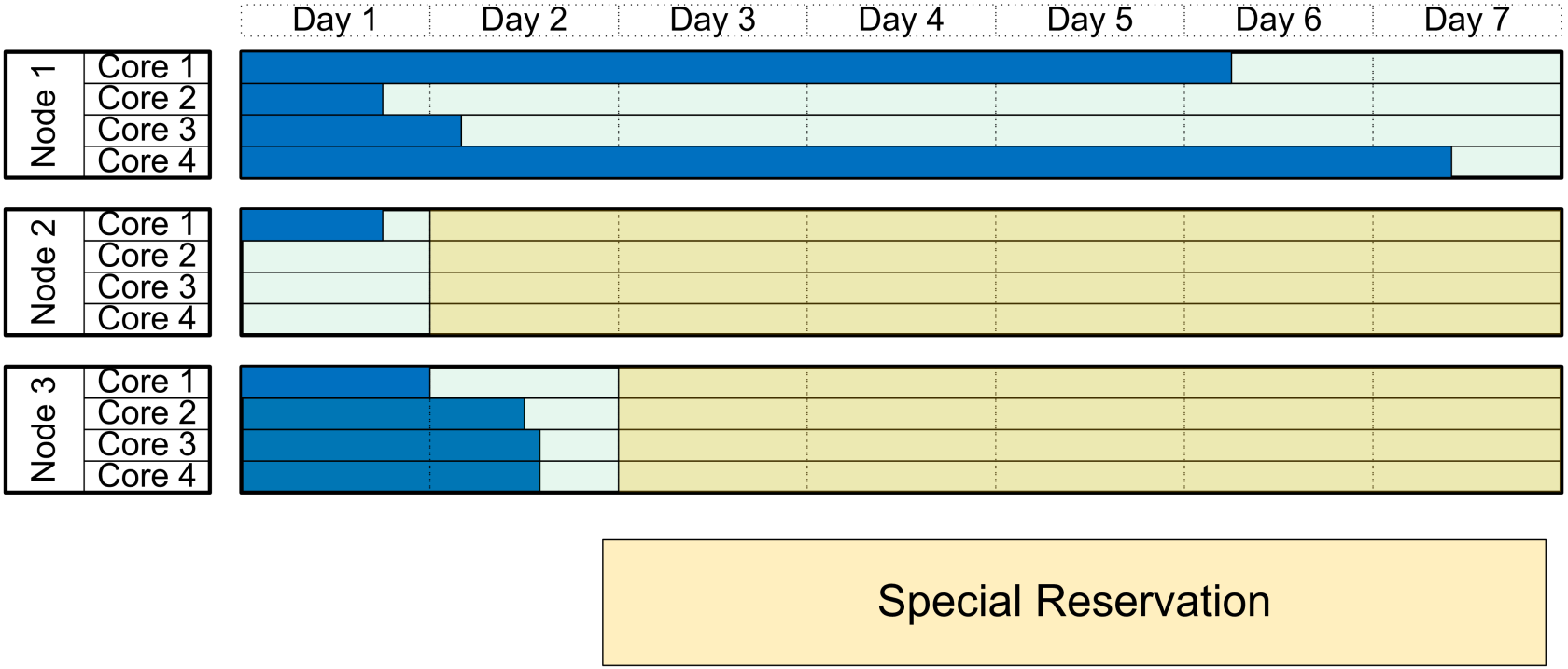
Topology

Usage limits on a cluster

There are 2 types of usage limits:

- Usage limits that prevent the scheduling system from being overloaded.
- Usage limits that prevent a user from monopolizing the cluster
  - by starting jobs on all resources of a cluster which will run for a long period of time.
  - By starting jobs that last a very long time

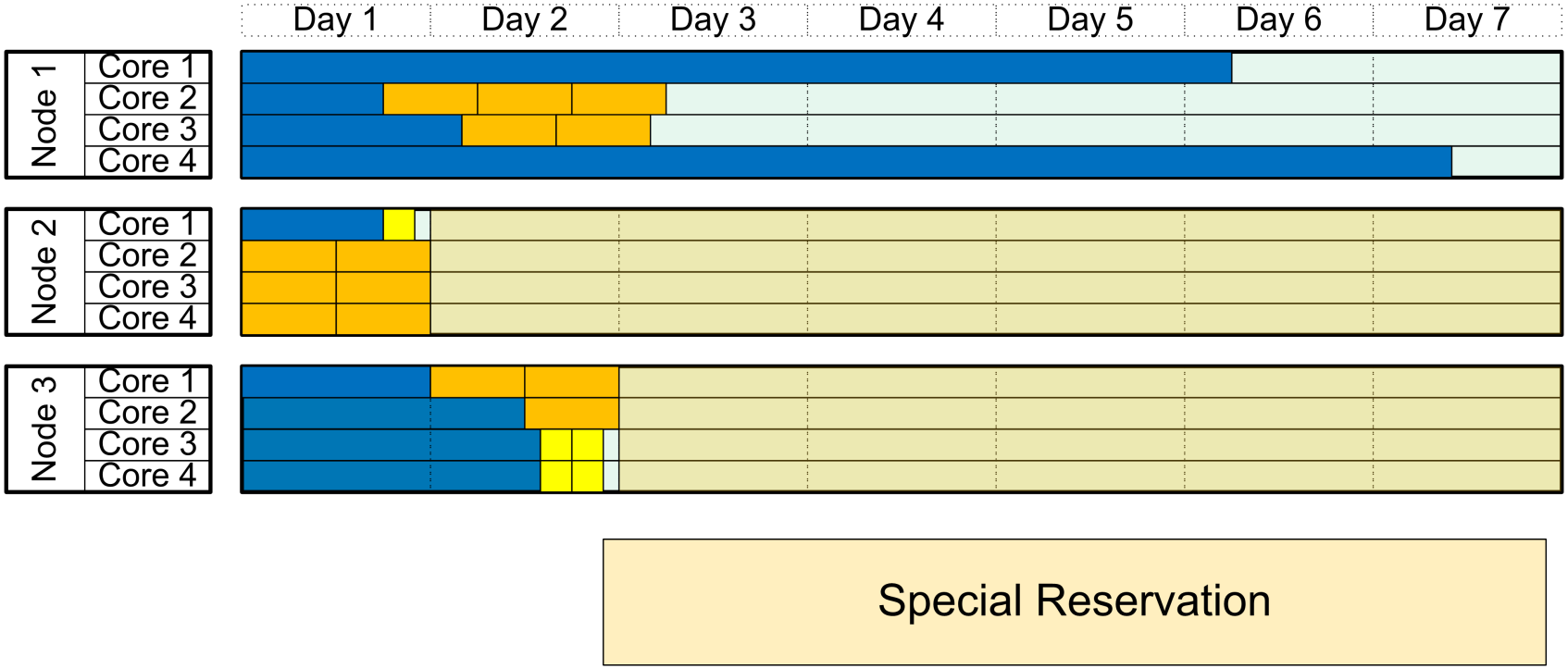
Reservations



Reservations

- Used for many purposes
  - Used to schedule outages: Security patch that requires an reboot
  - Used to reserve resources for special occasions, such as a workshop
  - Each job also creates reservations
- One can see reservations on a cluster via “scontrol show reservations”command

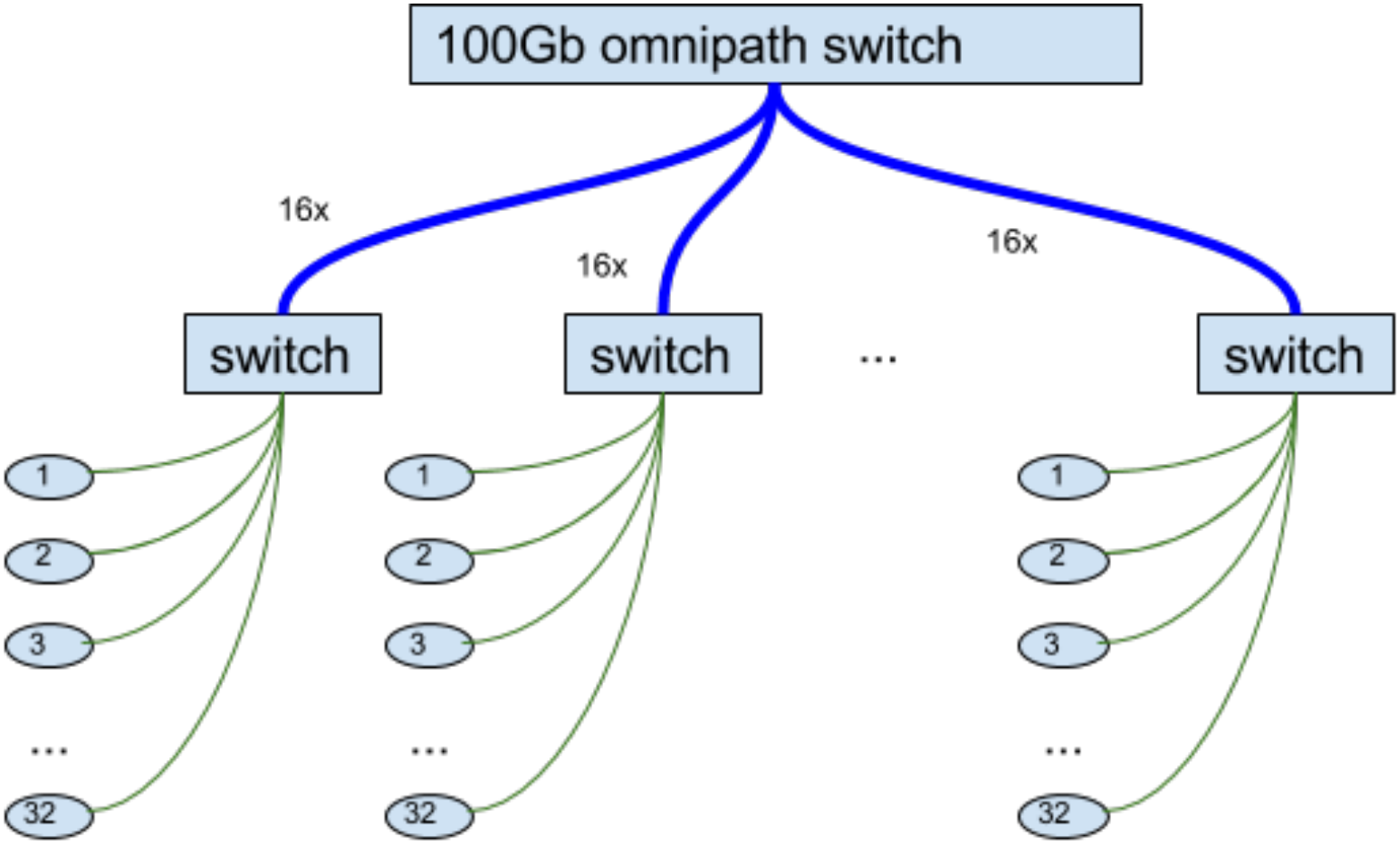
Reservations and short serial jobs



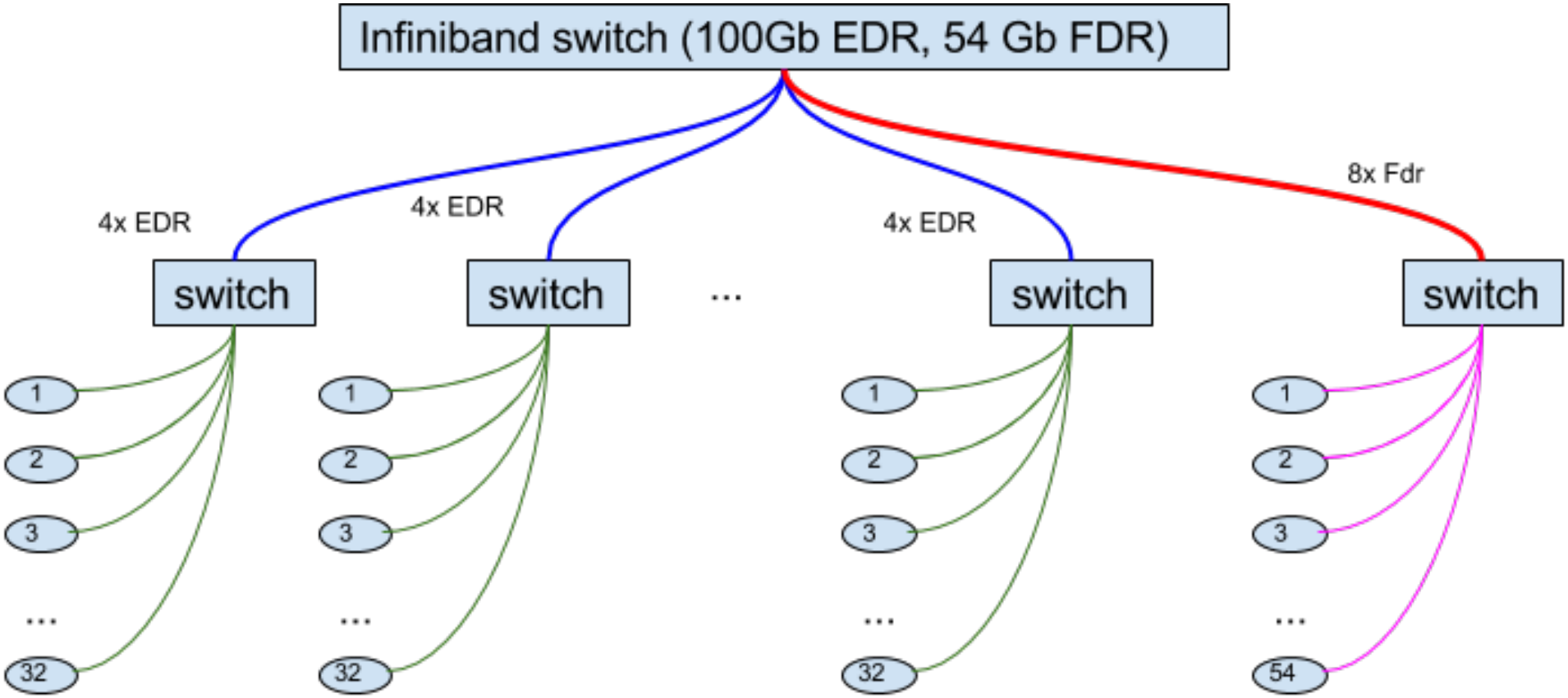
Topology

- On all but smallest cluster nodes some nodes are more closely connected than others.
- If your job can run on multiple resources it may be limited to closely connected nodes.
- This may mean while there are enough idle resources to start your job on the cluster and your job has the highest priority you job may not be scheduled due to not enough nodes being available that are closely connected.

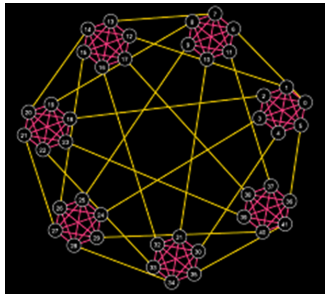
Topology on Cedar



Topology on Graham



Topology on Niagara



- Niagara is for large 1000 − 100,000 core jobs
- Niagara's topology is called Dragonfly+ with adaptive routing
- Beyond the scope of this presentation
- Contact support@compute.canada for help

Getting information on your Cluster

Sinfo -R

- Shows Nodes that are down and the reason why usually some error.

```
[kamil@cedar5 projects]$ sinfo  -R | head -12
REASON          USER    TIMESTAMP       NODELIST
Not responding     root    2017-06-23T14:10:54 cdr[137-139,147,270]
batch job complete f root    2017-08-20T05:36:07 cdr811
Not responding     slurm   2017-08-29T02:41:01 cdr119
Prolog error       root    2017-08-27T14:31:25 cdr47
batch job complete f root    2017-08-23T01:36:00 cdr52
batch job complete f root    2017-08-17T14:07:09 cdr[53,62]
 Epilog error       root    2017-07-25T16:39:47 cdr61
```

sinfo --states=idle

- Shows idle nodes and partitions (When a node is in multiple partitions it shows it multiple times)

```
kamil@cedar5 projects]$ sinfo --states=idle | head -15
PARTITION       AVAIL  TIMELIMIT  NODES  STATE NODELIST
cpubase_interac    up   12:00:00      7   idle cdr[552,556,682,693,695-696,848]
cpubase_bycore_b1  up    3:00:00     17   idle
cdr[358,362,365-367,369-374,377-379,381-382,384]
cpubase_bycore_b2  up   12:00:00      0    n/a
cpubase_bycore_b3  up 1-00:00:00      0    n/a
cpubase_bycore_b4  up 3-00:00:00      0    n/a
cpubase_bycore_b5  up 7-00:00:00      0    n/a
cpubase_bycore_b6  up 28-00:00:0      0    n/a
cpubase_bynode_b1* up    3:00:00     66   idle
cdr[358,362,365-367,369-374,377-379,381-382,384,391,413,497,501,504,51
0,542,555,560,563,568,579,598,600,612,615,626,631,644,648,652,654,657,6
67,669,676,684,711,716-717,721,724-725,729,731-732,735,739,744,758,761
,774,778,785,805-806,808,837,855]
```

Partition Stats

(CC script)

Node type	Max walltime					
	3 hr	12 hr	24 hr	72 hr	168 hr	672 hr
Number of Queued Jobs by partition Type (by node:by core)						
Regular	1:15	2:31	2:145	11:187	86:69	3:2
Large Mem	0:0	0:0	0:0	0:0	0:1	0:1
GPU	0:1	0:526	10:10	0:0	189:4	0:0
Number of Running Jobs by partition Type (by node:by core)						
Regular	60:6	4:2	45:836	5:90	11:1065	1:4
Large Mem	0:0	0:0	0:0	0:0	0:0	1:0
GPU	0:20	2:10	13:2	0:0	0:0	0:3
Number of Idle nodes by partition Type (by node:by core)						
Regular	0:0	0:0	0:0	0:0	0:0	0:0
Large Mem	3:1	0:0	0:0	0:0	0:0	0:0
GPU	17:1	11:1	0:0	0:0	0:0	0:0
Total Number of nodes by partition Type (by node:by core)						
Regular	851:411	821:391	756:346	636:276	180:100	90:50
Large Mem	27:12	24:11	24:11	20:3	3:2	2:1
GPU	156:78	144:72	116:58	104:52	13:12	13:12

Getting information on your Cluster

Command	What its used for
sinfo --states=idle	Show idle node on cluster
sinfo -R	Show down, drained and draining nodes and their reason
sinfo --Node --long	Show detailed node info.
scontrol show reservation	Shows reservations on the cluster
partition-stats	Compute Canada script to show jobs and nodes by partition

scontrol create reservation
user=root starttime=now
duration=infinite
flags=maint
nodes=<nodeid>

Cluster information

BREAK FOR PRACTICE

Why does my job not run?

- List of reasons your job is not running in order of probability.
  1. There is a problem with the job
  2. The Job is blocked
  3. Other jobs have greater priority
  4. Resources are not available
  5. There is a problem with the scheduling system or cluster.

Common Problems

- The Job request more resources than are available on the system or node or practical to run on the system.
- ex)
  - You can request 10,000 cores on cedar
  - Request more than 3TB of RAM per node
  - Request 5 nodes each with 2TB per node

Problem with my job

1. Is the Job blocked? “squeue −u <user name>”
   – Find out more? “scontrol show jobid -dd <jobid>”
2. Is the Job on hold? Are there dependencies?

Is there a problem with my job?

3. What is my jobs priority? Compare it to other jobs on cluster run: “sprio”

   If you have much lower priority find out why:

   use: “sshare”

   - Wait until priority improves over time.
   - Ask fellow group members to run less.
   - Ask for your professor to apply for a RAC allocation.

Is there a problem with the cluster?

4. If you have high priority and your job is queued check to see if the resources are available
   a. Use “partition-stats” to see if there are enough resources available on enough nodes to start your job. Check the WestGrid webpage to see if there is an outage scheduled.

Is there a problem with cluster

5. Is there a reservation or system outage
   - Check the Compute Canada webpage / MOTD on the system to see if there is an outage scheduled.
   - Check for an reservation on the system “scontrol show reservation”

Send email to support@computecanada.ca

- Make sure you always include the following at the beginning of the email
  - Name of the cluster, jobid , userid
  - The location of the jobscript you submitted.
  - Any output or error of the job run.
  - Also make sure the name of the cluster is in the subject, ex: “job 123456 fails to run on the Cedar cluster”
- Brief but complete description of the problem.
- You should try to include the output of any commands like those descripted in the talk earlier. Please include any output of commands that you have run which convinced you there is a problem. A lot of these commands give the state of the job or cluster at the moment and this way we can analyze the situation as you saw it.

Scheduling in the future

- Many more levels of topology
- Enforcing exclusivity with granularity
- Data movement, backups, recovery, latency, bandwidth, move job to data not data to job.
- Failure tolerant jobs and scheduling
- Power aware jobs and scheduling
- Scheduling provisioning of nodes
- Scheduling VMs and containers.
- Cloud /Grid scheduling including both batch jobs and services on the same system, virtual network management, all the points above in a integrated system

QUESTIONS?