

Jobs – from *one* to *many*



Promise: You will learn techniques to scale up work + some other useful things.

Prerequisites: UNIX shell, editor, basic knowledge of jobs & Slurm

Me

- 10 years research in computer science
- 10 years HPC sysadmin
- Projects: {N,E}ESSI & LUMI User Support



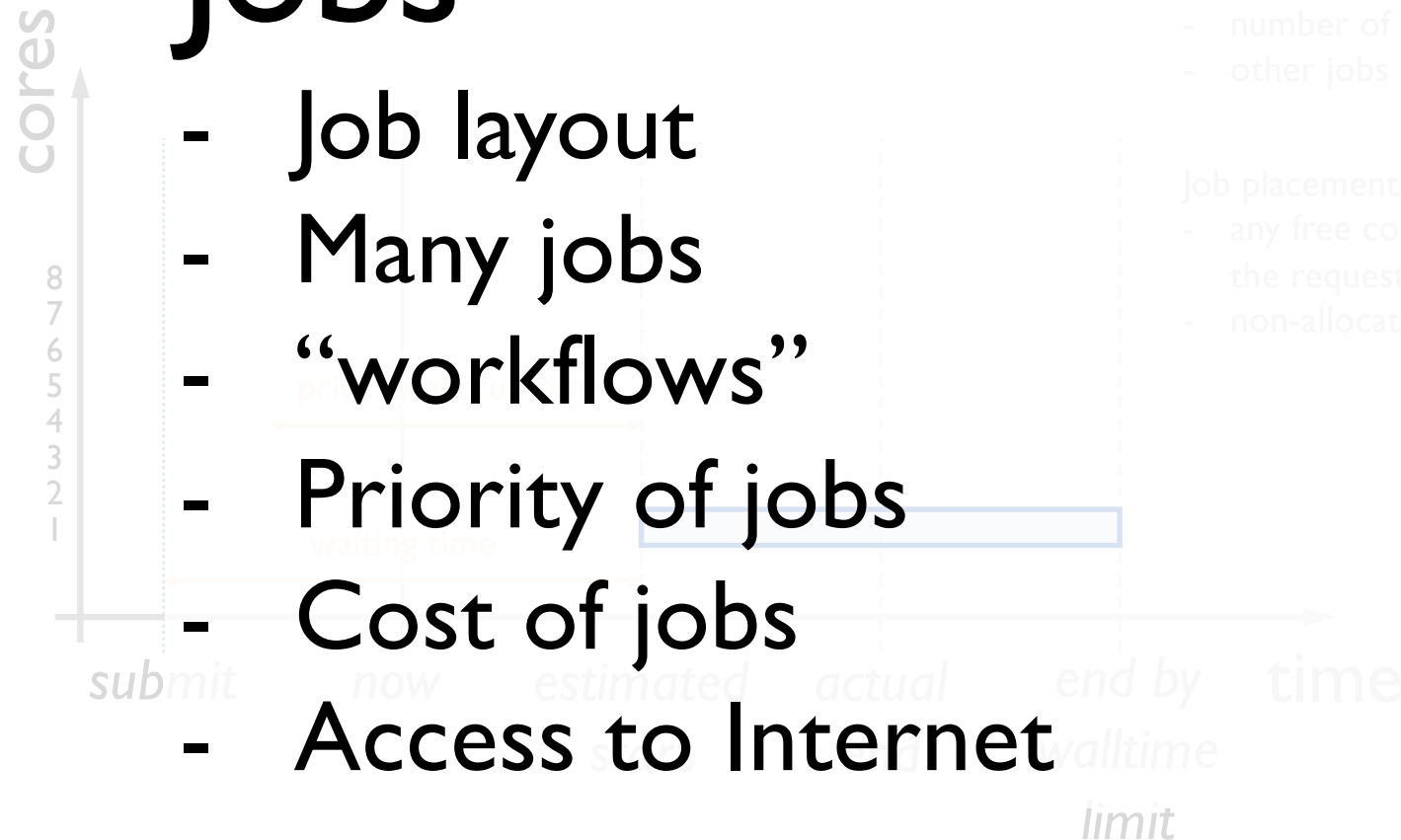
Waiting time factors:

- job size (#cores, RAM, wall lim, QoS)
- number of jobs (this user, other users)
- other jobs

Job placement:

- any free core on any compute node in the requested partition
- non-allocated memory \leq job mem req

Jobs



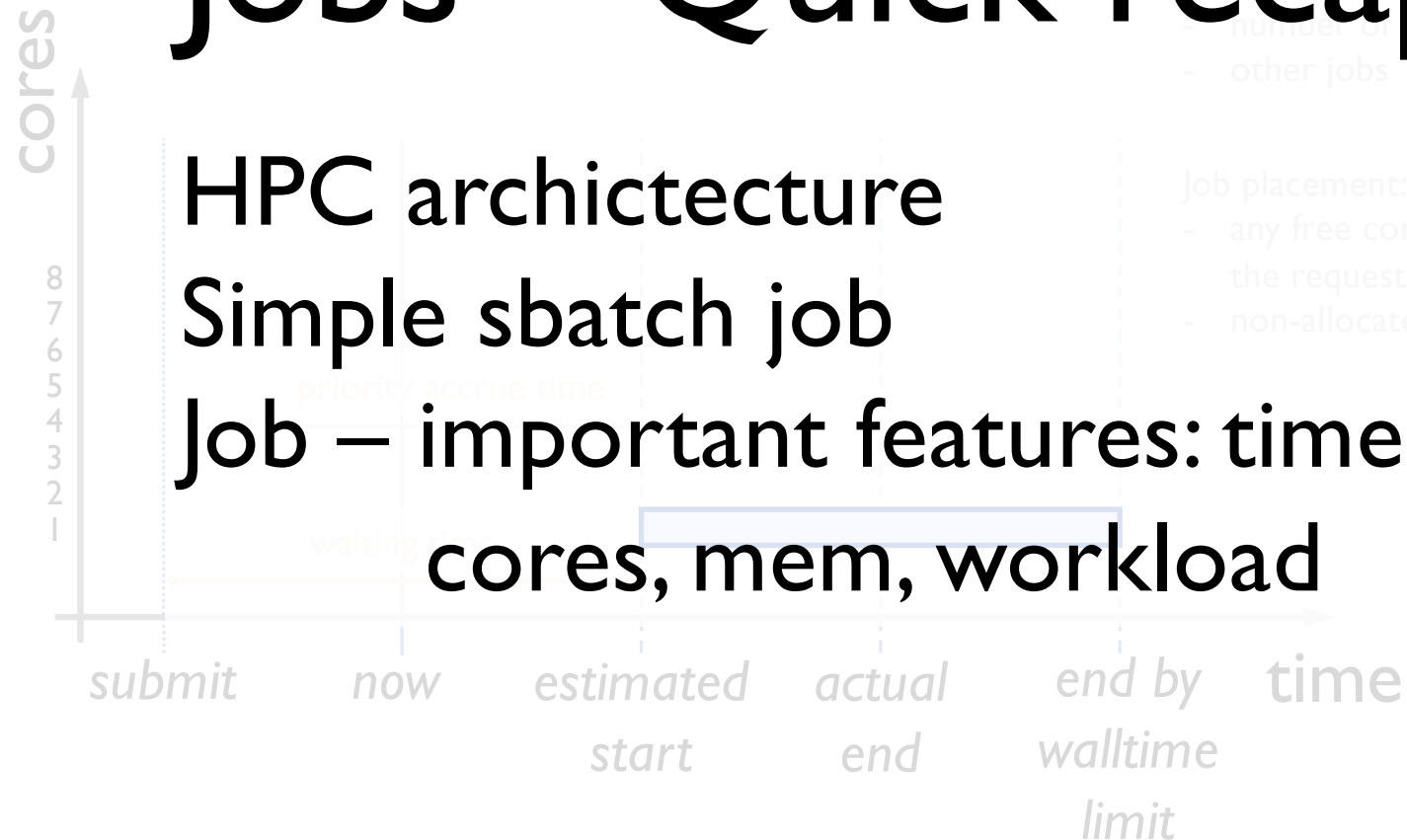
Waiting time factors:

- job size (#cores, RAM, wall lim, QoS)
- number of jobs (this user, other users)
- other jobs

Job placement:

- any free core on any compute node in the requested partition
- non-allocated memory \leq job mem req

Jobs – Quick recap



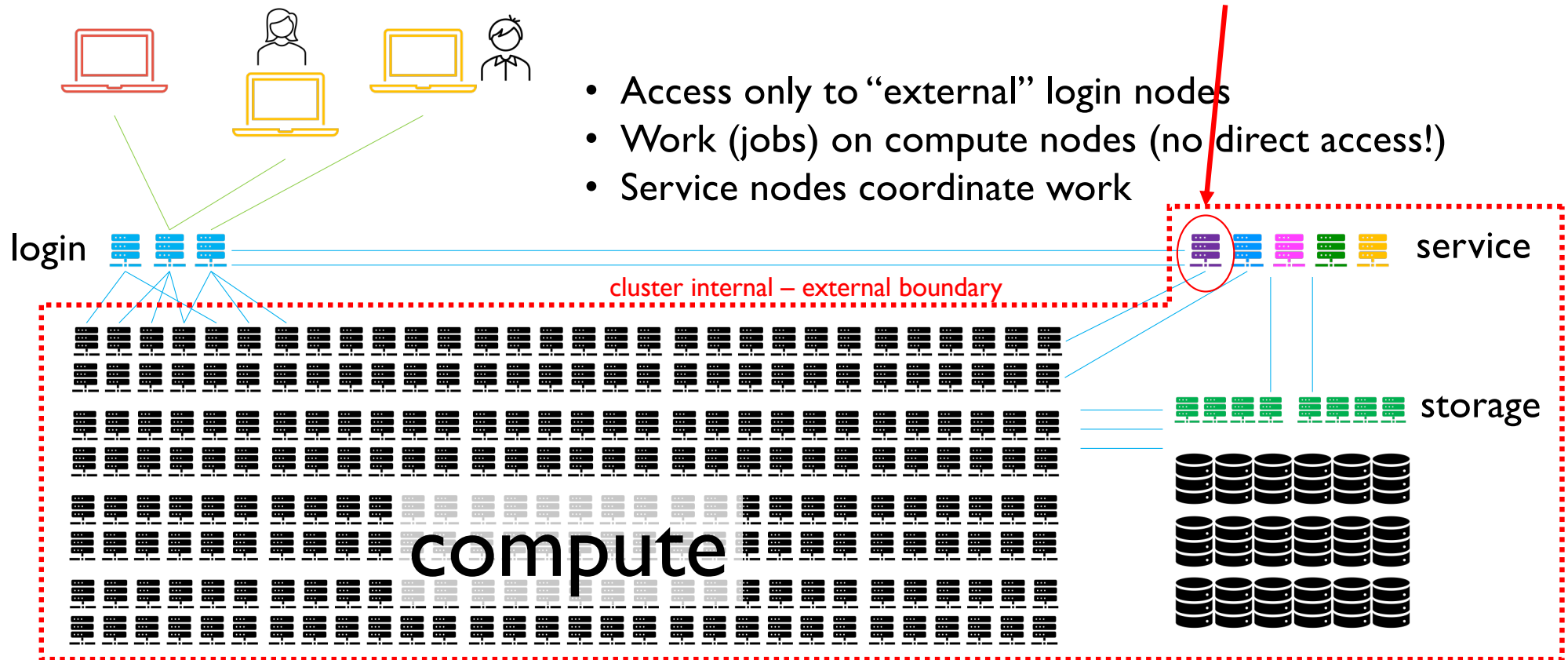
Waiting time factors:
- size of job
- RAM, wall lim, QoS
- number of cores (this user, other users)
- other jobs

Job placement:
- any free core on any compute node in the requested partition
- non-allocated memory \leq job mem req

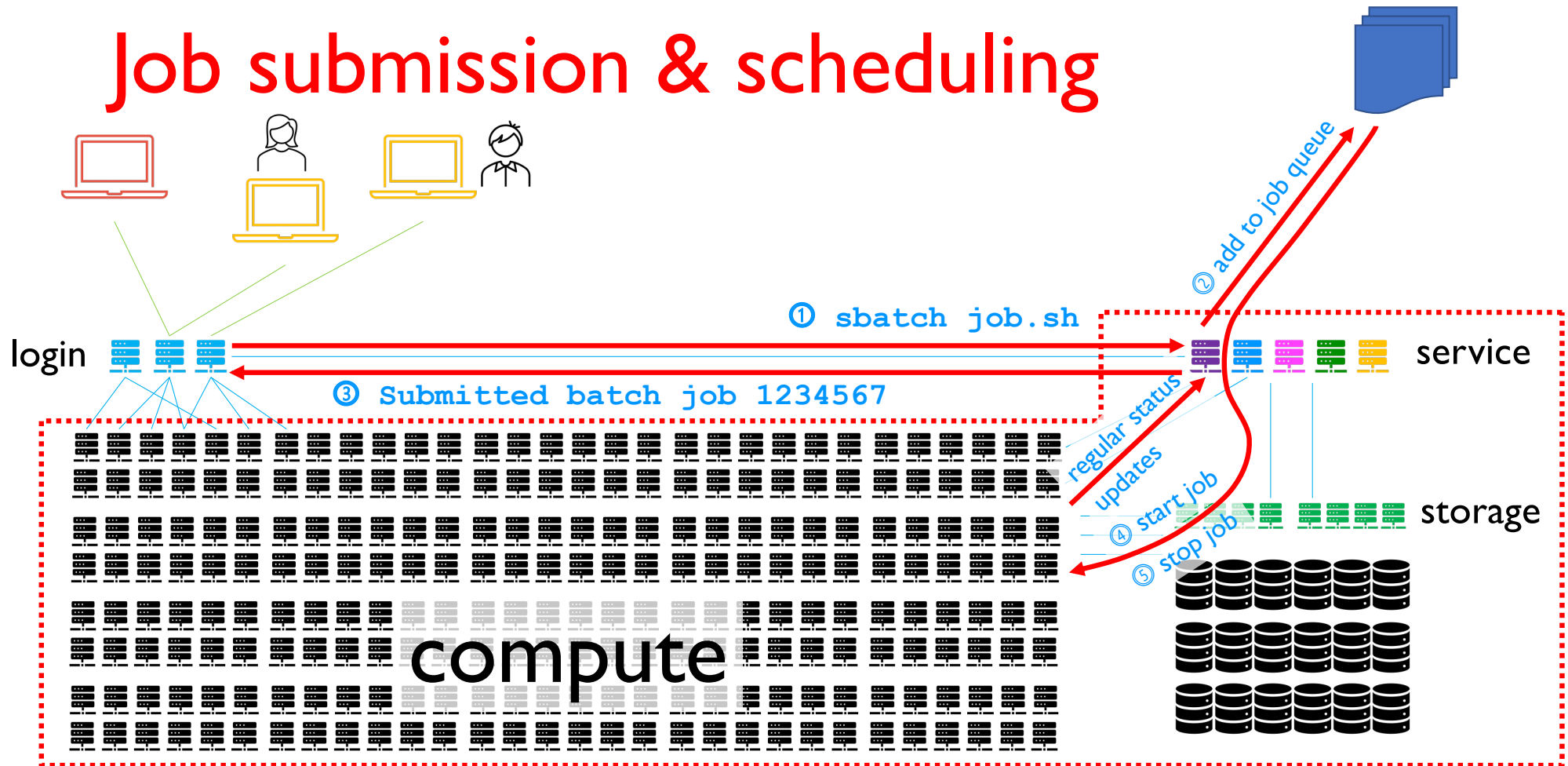
HPC architecture

Runs SLURM

- Access only to “external” login nodes
- Work (jobs) on compute nodes (no direct access!)
- Service nodes coordinate work



Job submission & scheduling



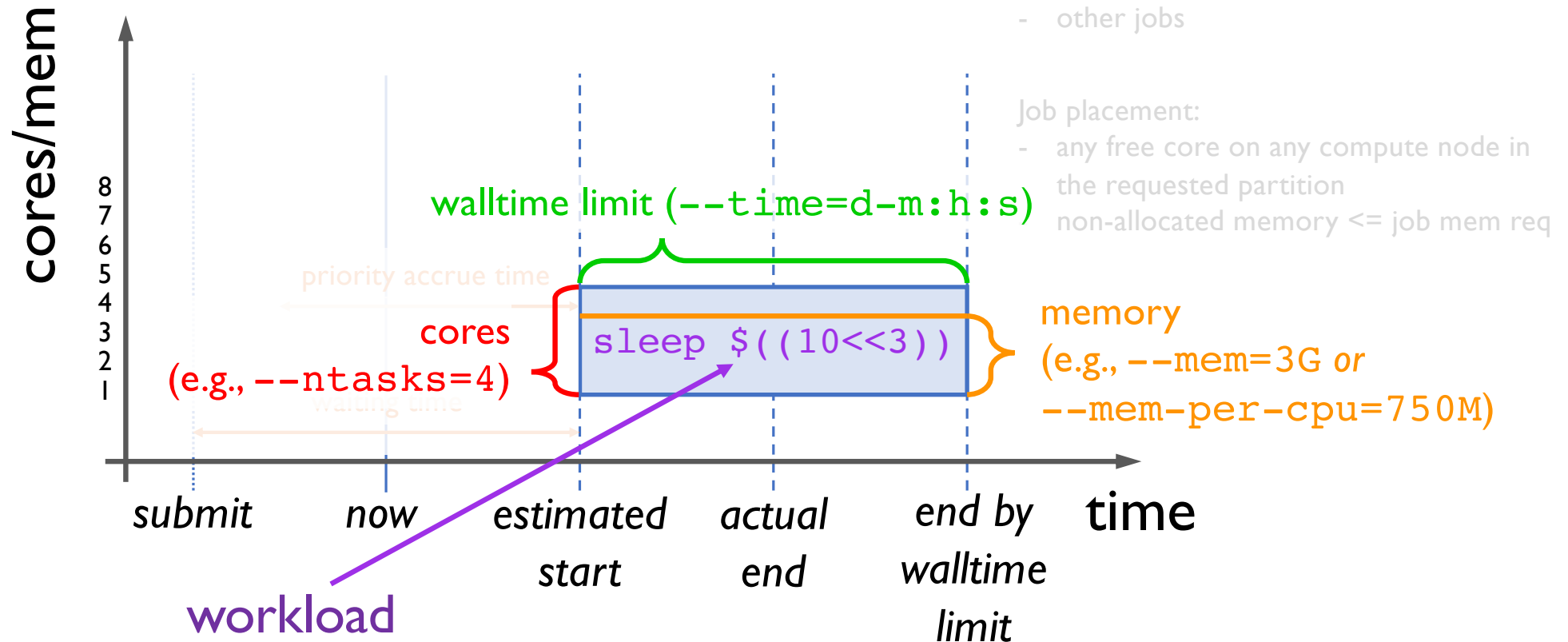
Jobs – important features

Waiting time factors:

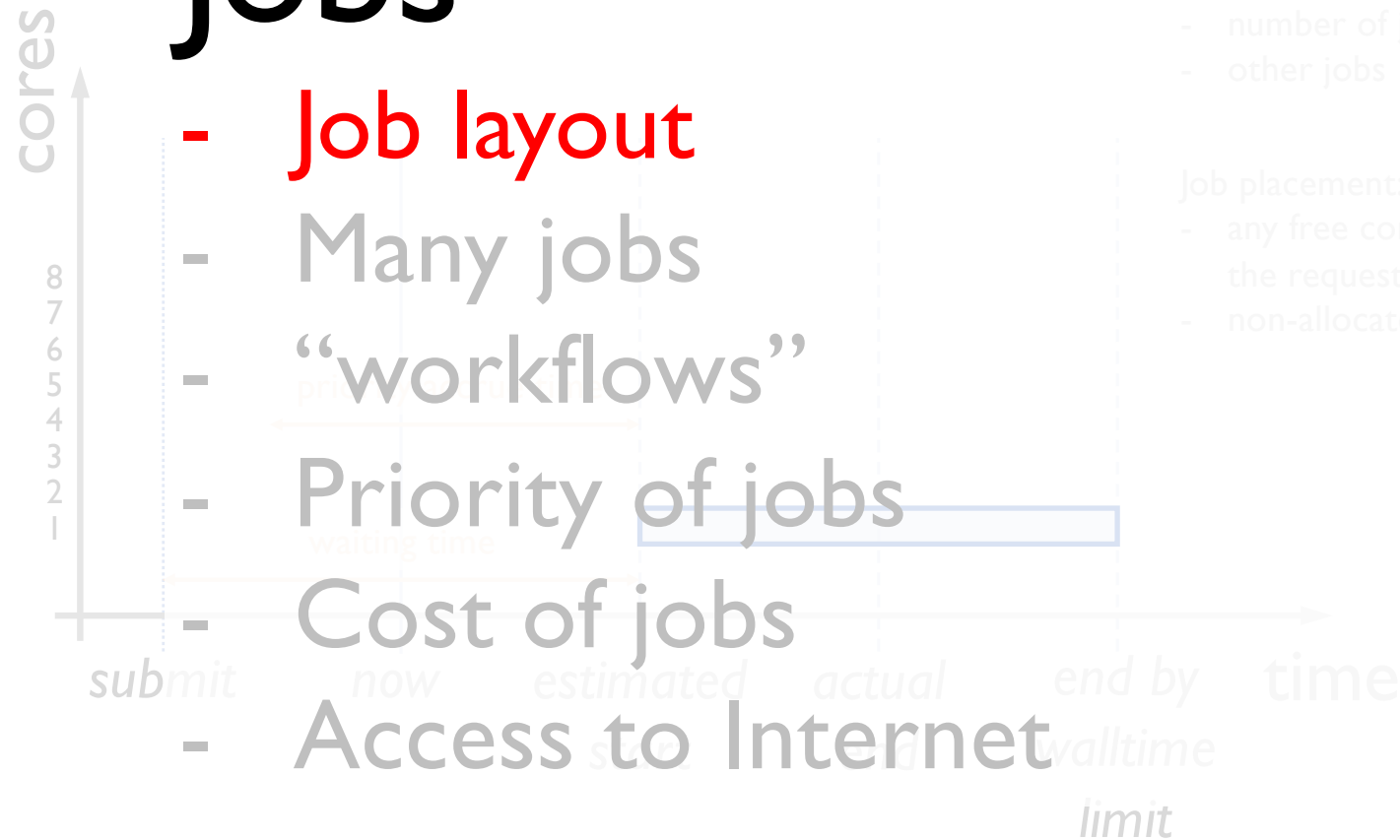
- job size (#cores, RAM, wall lim, QoS)
- number of jobs (this user, other users)
- other jobs

Job placement:

- any free core on any compute node in the requested partition
- non-allocated memory \leq job mem req



Jobs



Waiting time factors:

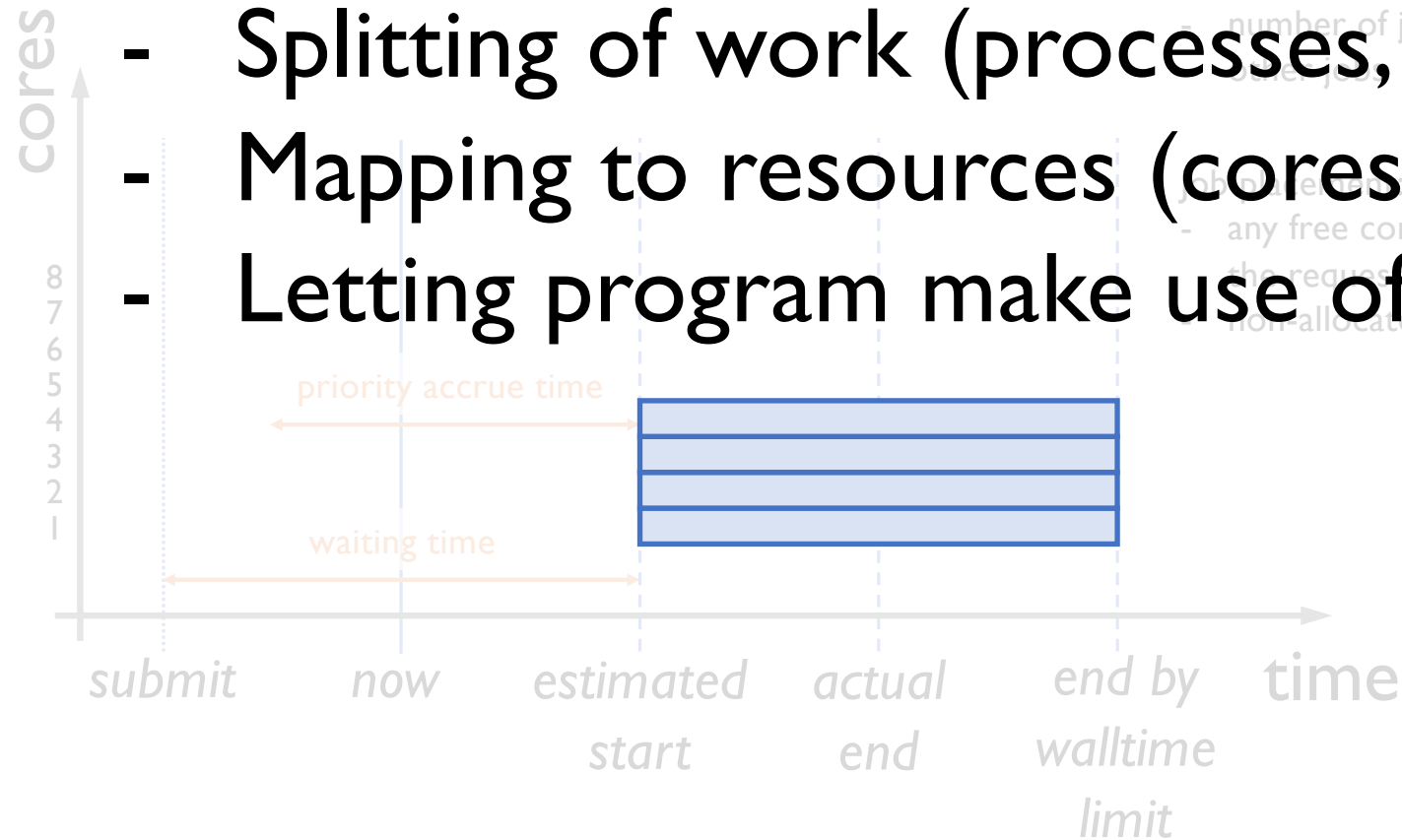
- job size (#cores, RAM, wall lim, QoS)
- number of jobs (this user, other users)
- other jobs

Job placement:

- any free core on any compute node in the requested partition
- non-allocated memory \leq job mem req

Job layouts

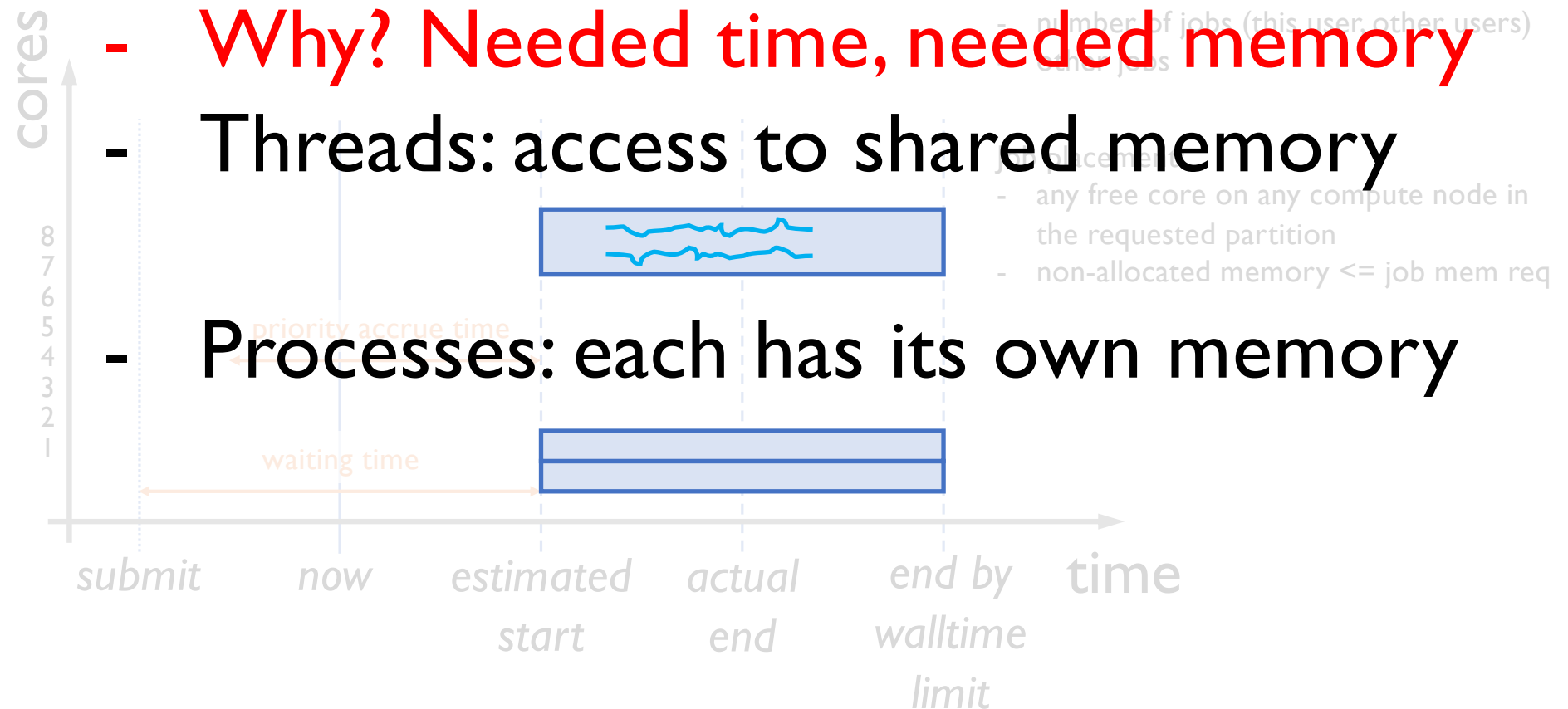
- Splitting of work (processes, threads)
- Mapping to resources (cores, nodes)
- Letting program make use of layout



Waiting time factors:

- job size (#cores, RAM, wall lim, QoS)
- number of jobs (this user, other users)
- any free core on any compute node in the requested partition
- non-allocated memory < job mem req

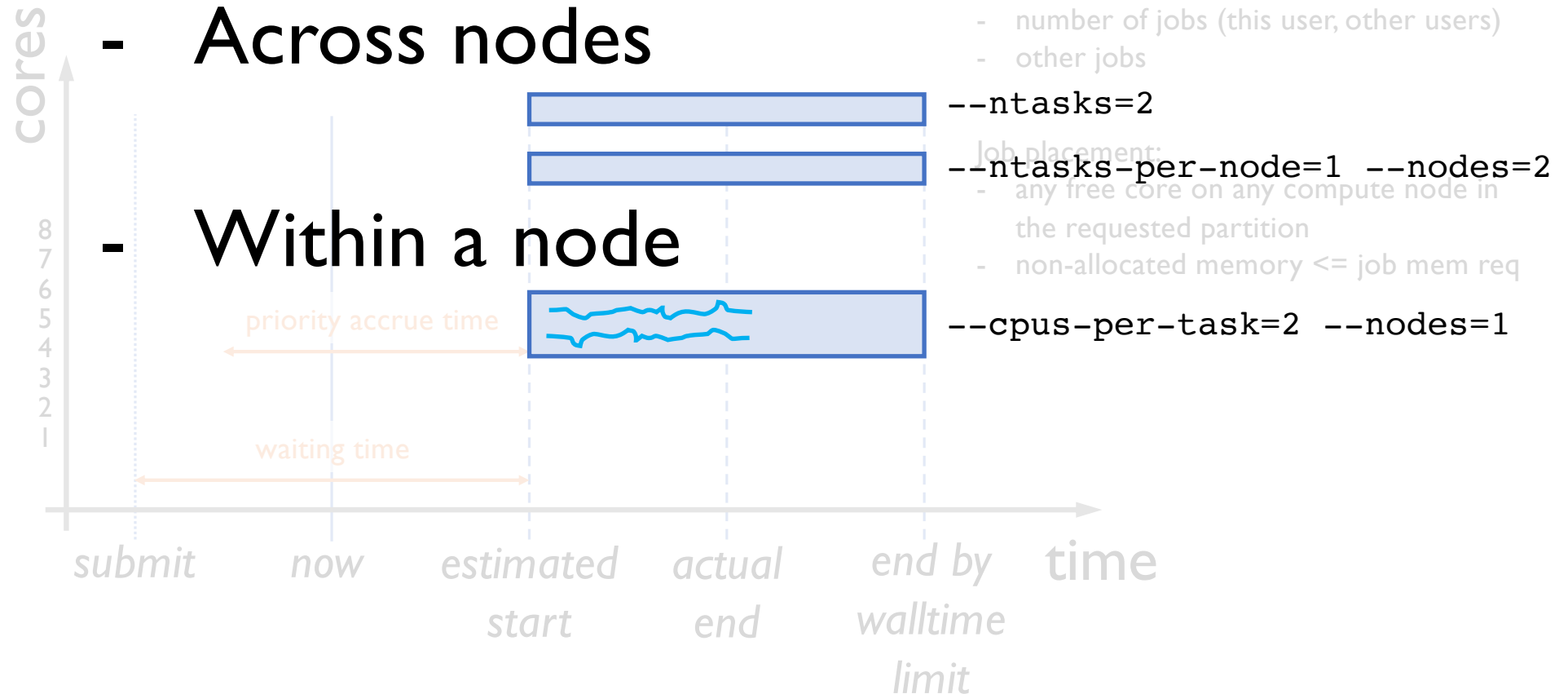
Job layouts – splitting of work



Job layouts – Mapping to resources

- Across nodes

- Within a node

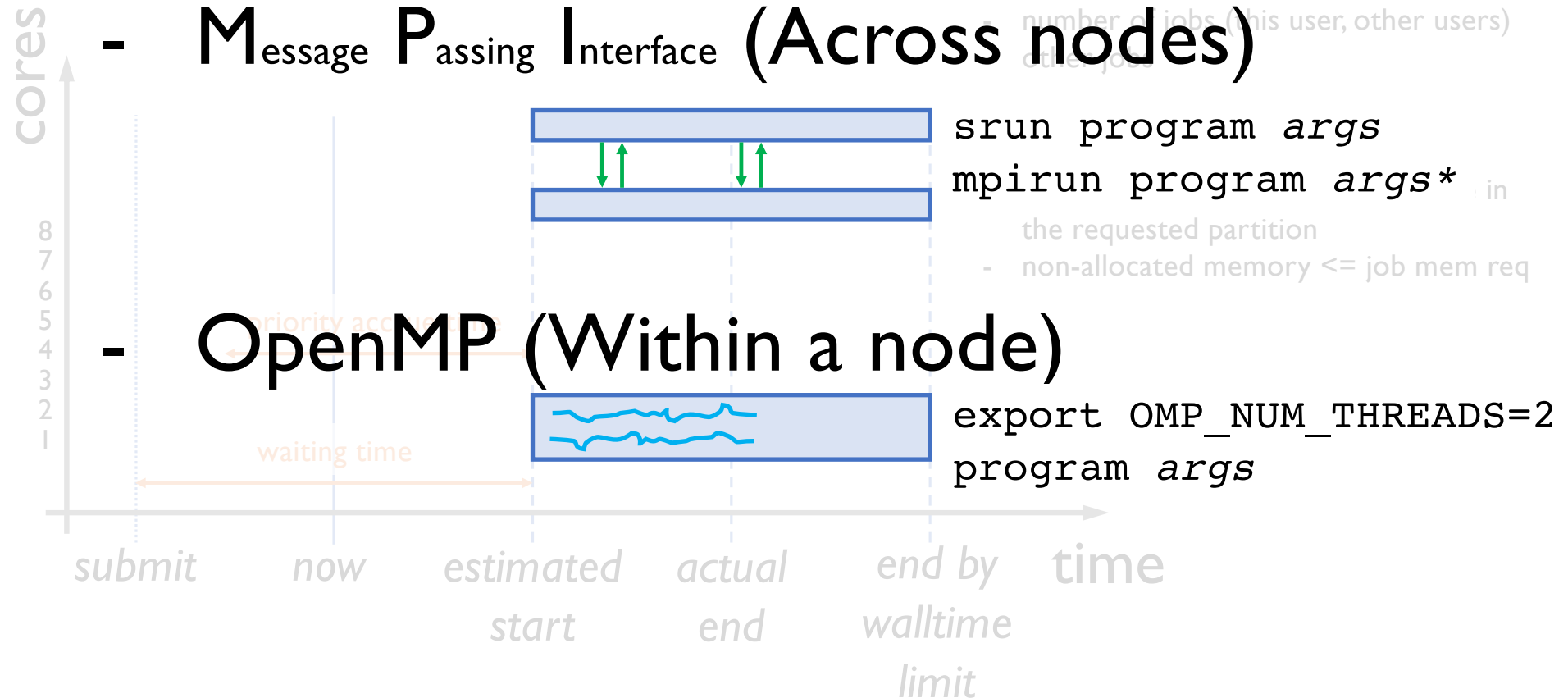


- job size (#cores, RAM, wall lim, QoS)
- number of jobs (this user, other users)
- other jobs

- job placement:
- any free core on any compute node in the requested partition
 - non-allocated memory \leq job mem req

Job layouts – Let a program use layout

- Message Passing Interface (Across nodes)



(*) We recommend to use srun. See documentation at https://documentation.sigma2.no/jobs/guides/running_mpi_jobs.html?highlight=mpirun

Job layouts – Examples

- Job uses 4 threads in 1 process (task)

...

#SBATCH --ntasks=1

#SBATCH --cpus-per-task=4

...

program *args*

...

Waiting time factors:

- job size (#cores, RAM, wall lim, QoS)
- number of jobs (this partition)
- the job's priority

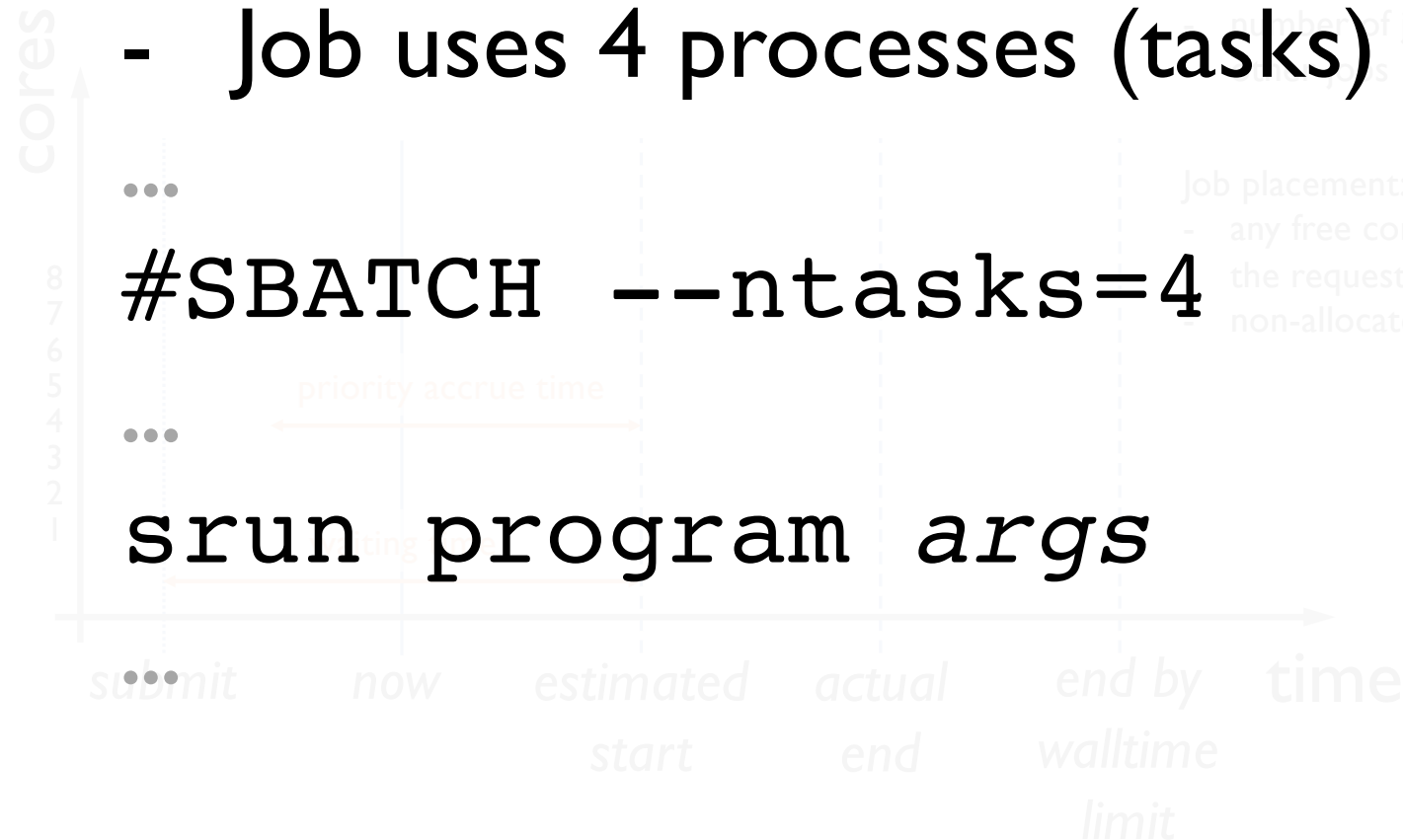
Job placement:

- any free core on any compute node in the requested partition
- non-allocated memory <= job mem req



Job layouts – Examples

- Job uses 4 processes (tasks) *anywhere*



Waiting time factors:

- job size (#cores, RAM, wall lim, QoS)

- number of jobs (this is not a factor)

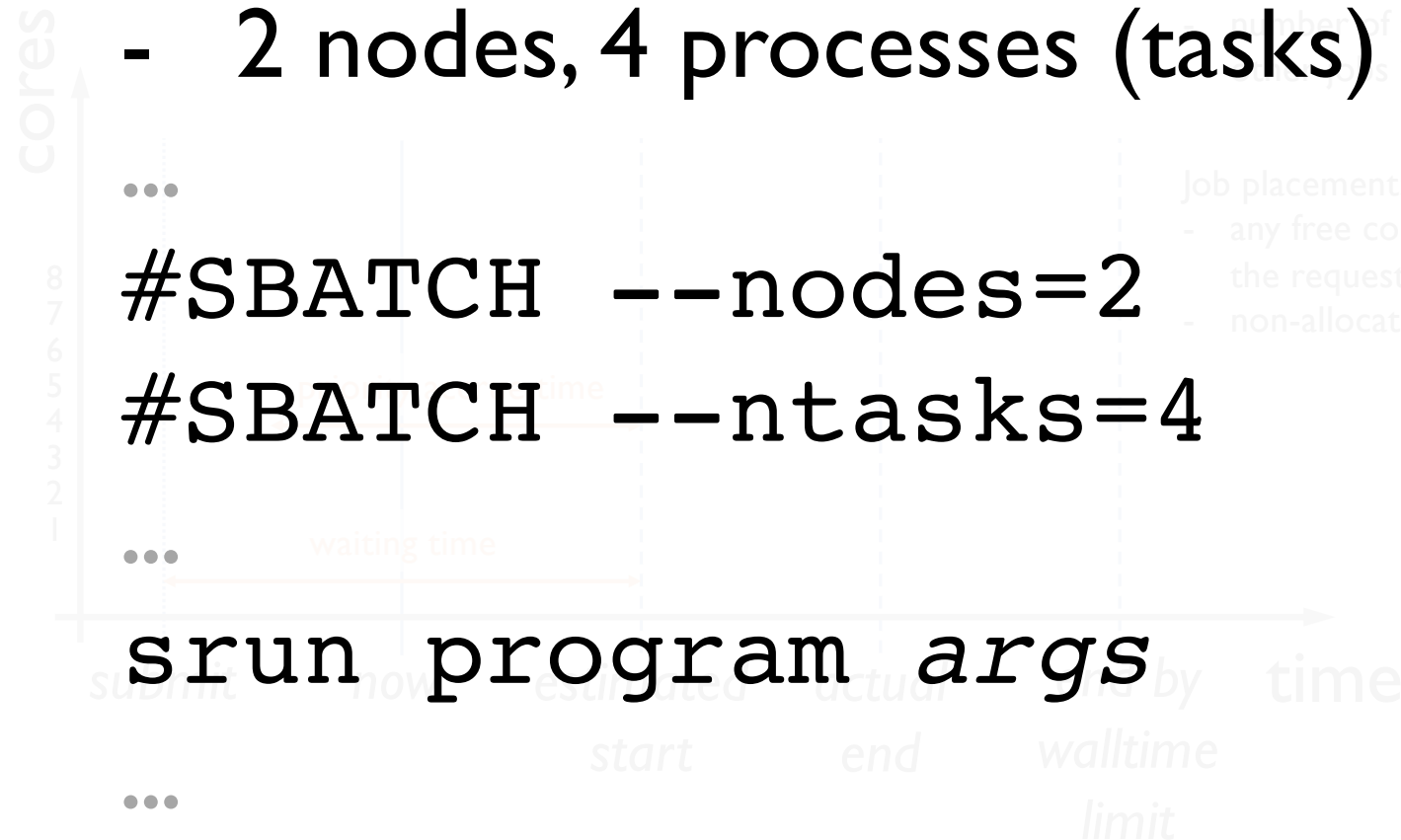
Job placement:

- any free core on any compute node in the requested partition

- non-allocated memory \leq job mem req

Job layouts – Examples

- 2 nodes, 4 processes (tasks)



Waiting time factors:

- job size (#cores, RAM, wall lim, QoS)
- number of jobs (this user, other users)

Job placement:

- any free core on any compute node in the requested partition
- non-allocated memory \leq job mem req

Job layouts – Examples

- 2 nodes, 2 processes (tasks) on each

...

```
#SBATCH --nodes=2
```

```
#SBATCH --ntasks-per-node=2
```

...

```
srun program args
```

...

Waiting time factors:

- job size (#cores, RAM, wall lim, QoS)
- number of jobs (this user only, users)

Job placement:

- any free core on any compute node in the requested partition
- non-allocated memory \leq job mem req

cores
8
7
6
5
4
3
2
1

submit

move

estimated

actual

by

time

start

end

walltime

limit

Job layouts – Examples

- 4 processes (tasks), 2 on each node

...

```
#SBATCH --ntasks=4
```

```
#SBATCH --ntasks-per-node=2
```

...

```
srun program args
```

...

Waiting time factors:

- job size (#cores, RAM, wall lim, QoS)

- number of jobs (this user, other users)

Job placement:

- any free core on any compute node in

- the requested partition

- non-allocated memory <= job mem req

cores
8
7
6
5
4
3
2
1

submit

move

estimated

actual

by

time

start

end

walltime

limit

Job layouts – What to use & when?

cores
8
7
6
5
4
3
2
1

- Computing effort

- How long does it run? test small

- Problem size

- How much memory? test small, read doc

- Program capabilities

- Can use multiple cores? read doc

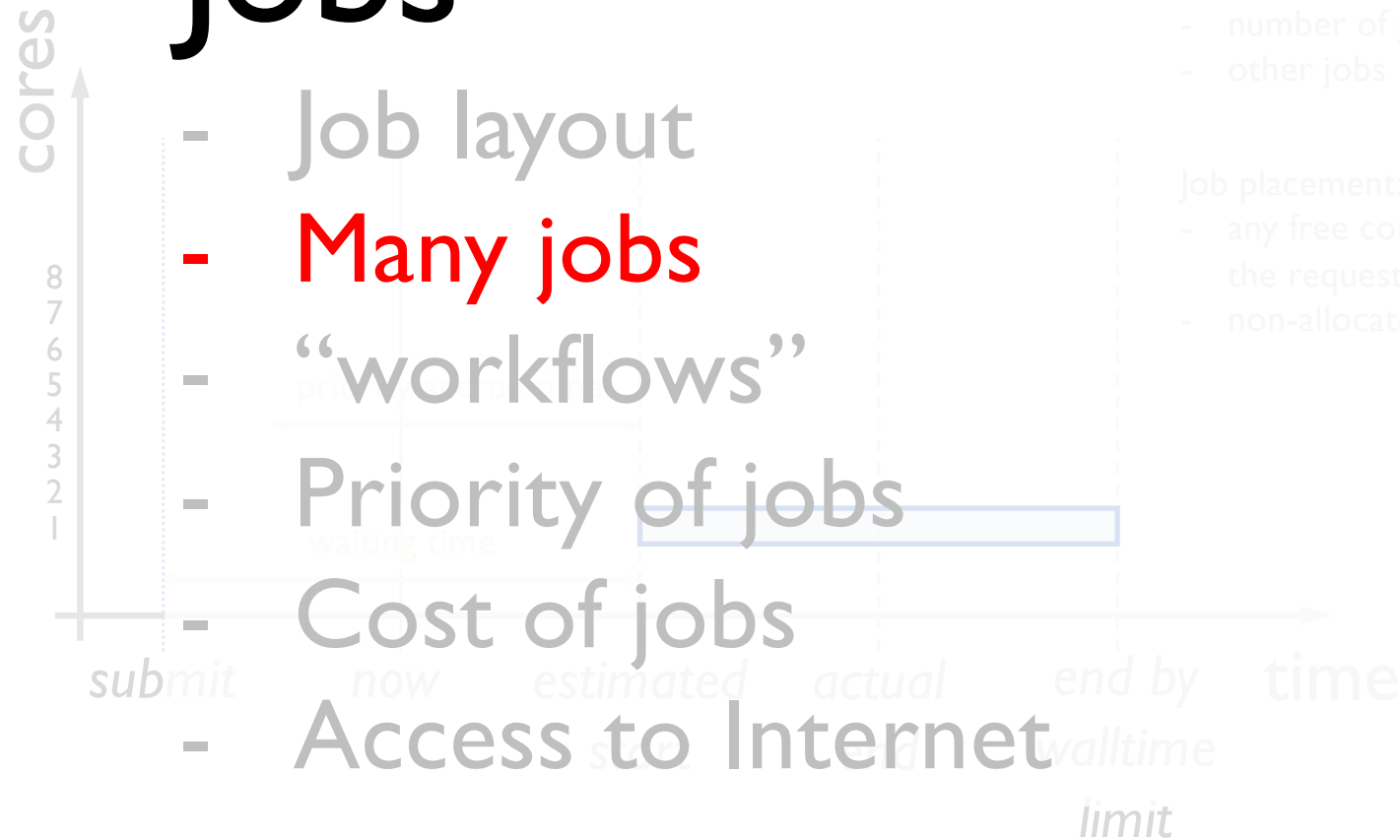
- OpenMP, MPI or both? read doc

- If possible (time, mem) run sequential.

- Either OpenMP or MPI. Lastly, both.

Start simple and small, then increase complexity (parallelism) and size

Jobs



Waiting time factors:

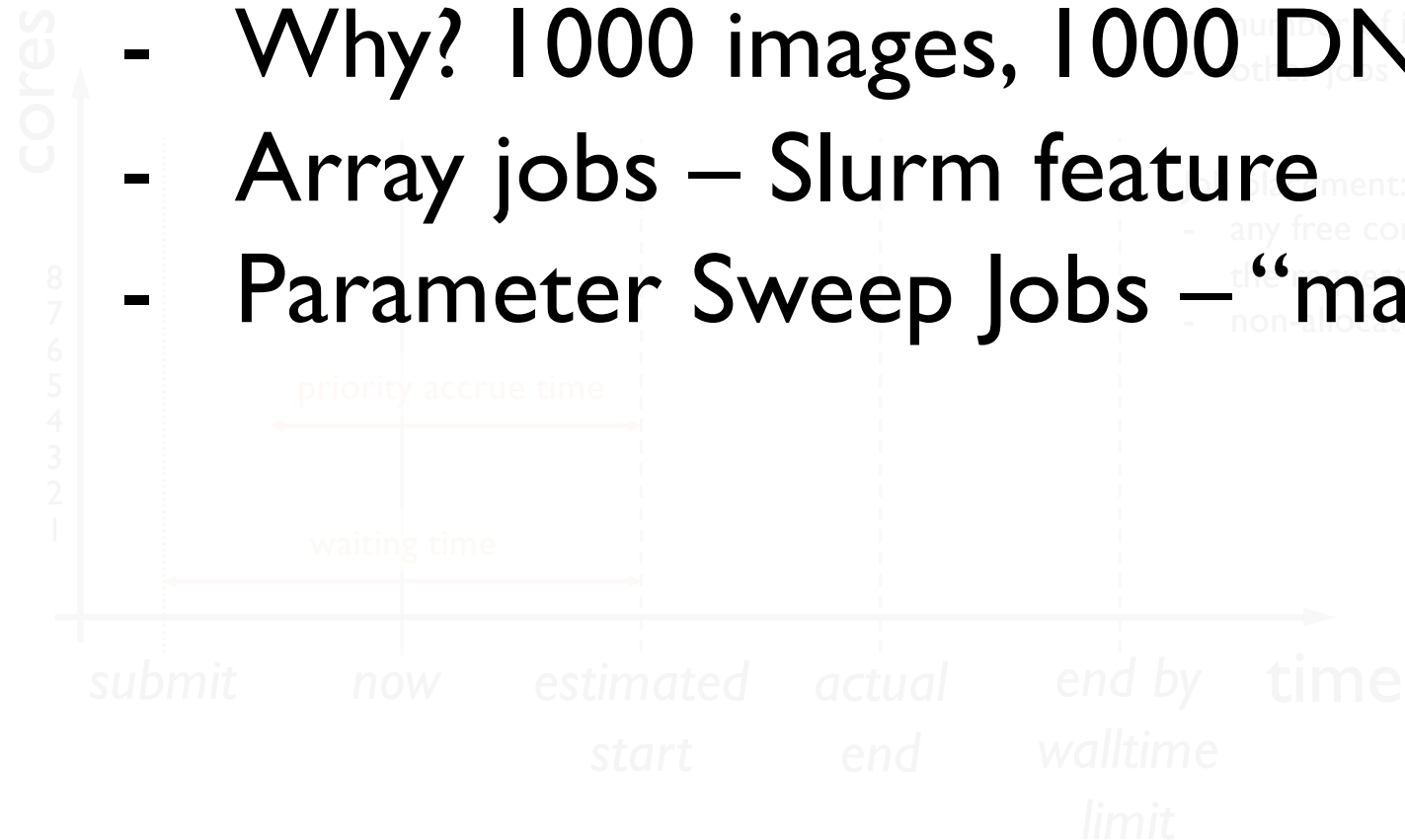
- job size (#cores, RAM, wall lim, QoS)
- number of jobs (this user, other users)
- other jobs

Job placement:

- any free core on any compute node in the requested partition
- non-allocated memory \leq job mem req

Many jobs

- Why? 1000 images, 1000 DNA seqs, ...
- Array jobs – Slurm feature
- Parameter Sweep Jobs – “manually”



Many jobs – Array jobs

```
#SBATCH --array=1-10
```

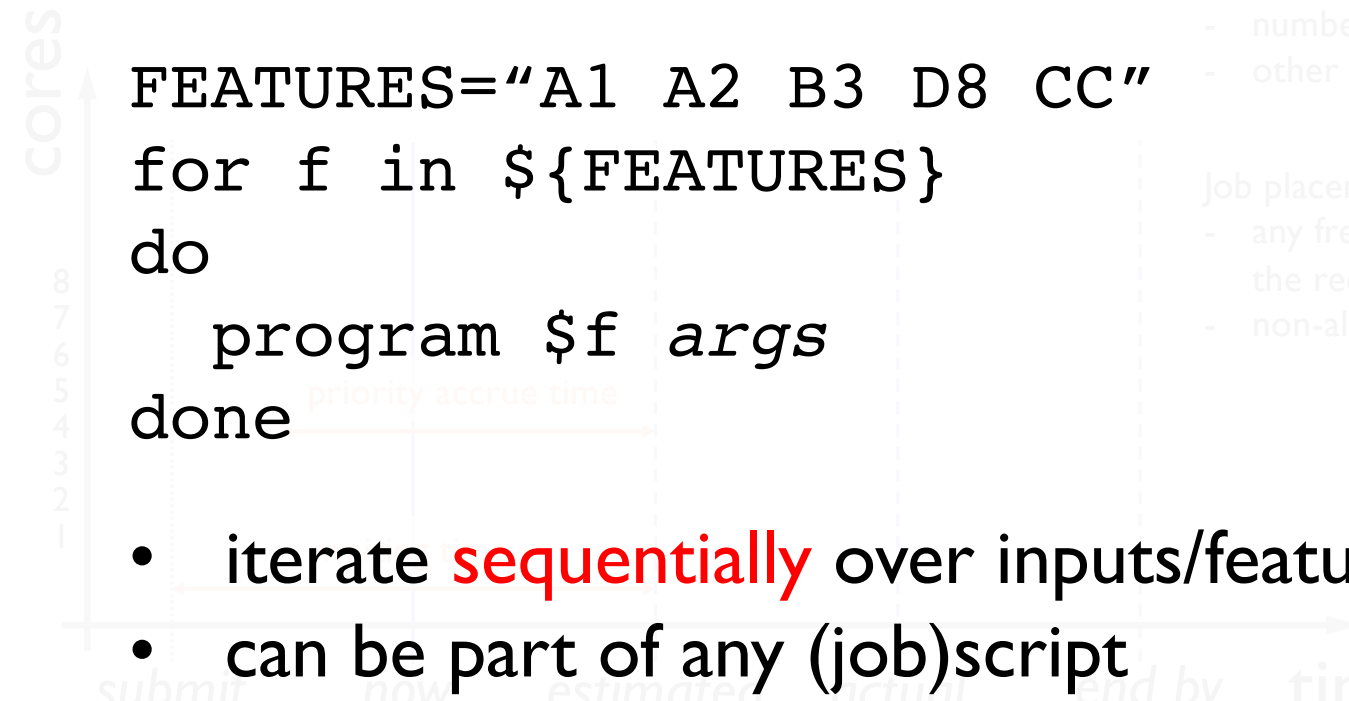
...

```
program ${SLURM_ARRAY_TASK_ID} args
```

- only one jobscript & sbatch call necessary
- squeue will show array workers as n_1, n_2,...
- max array size: `scontrol show config | grep -i array`
- program can make use of index(ed data)
- When? workload easily splitable and enumerable

https://documentation.sigma2.no/jobs/job_scripts/array_jobs.html

Parameter Sweep Jobs



```
FEATURES="A1 A2 B3 D8 CC"
for f in ${FEATURES}
do
    program $f args
done
```

- iterate **sequentially** over inputs/features
- can be part of any (job)script
- nested loops (combinations of features possible)
- When? workload not easily enumerable

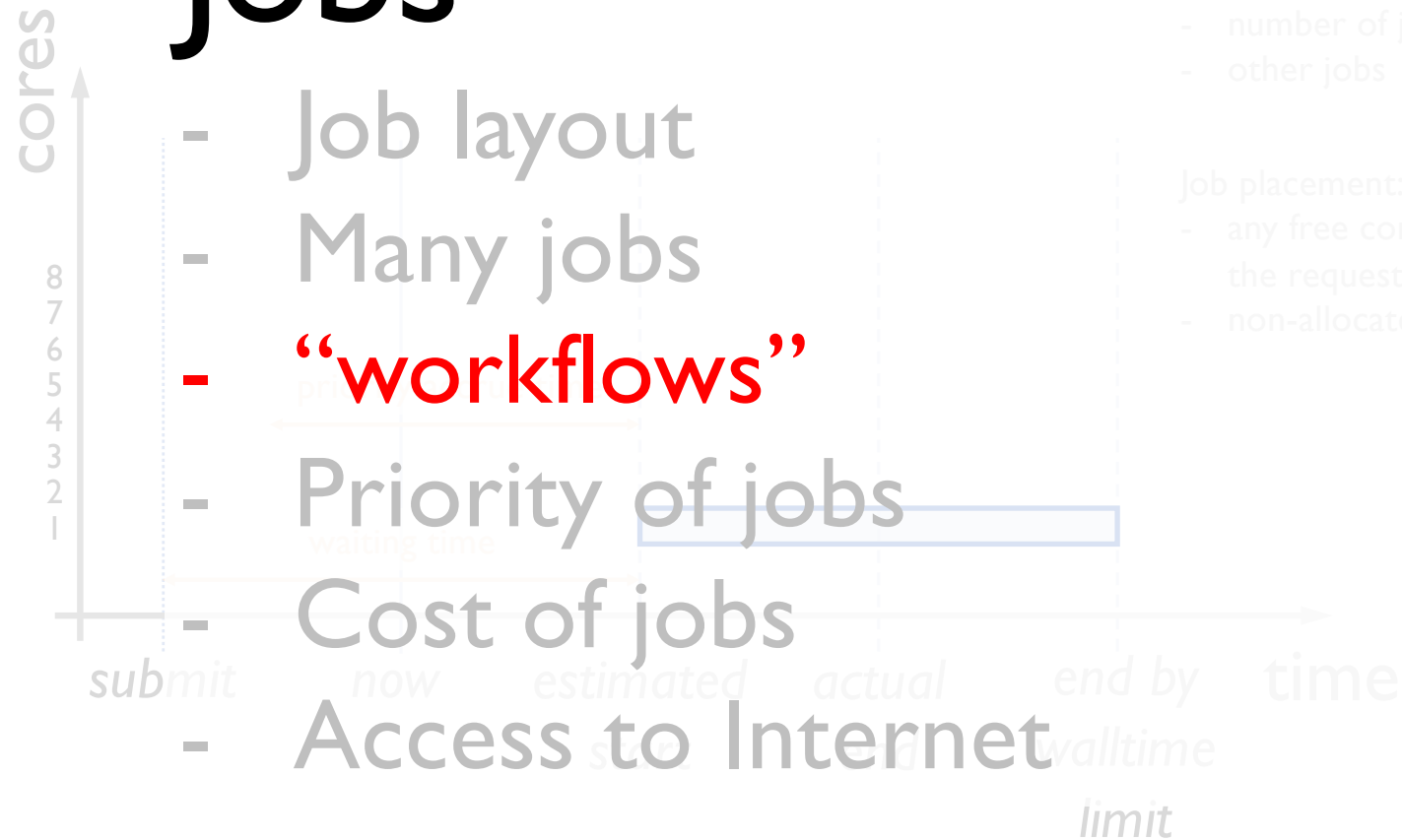
Waiting time factors:

- job size (#cores, RAM, wall lim, QoS)
- number of jobs (this user, other users)
- other jobs

Job placement:

- any free core on any compute node in the requested partition
- non-allocated memory \leq job mem req

Jobs



Waiting time factors:

- job size (#cores, RAM, wall lim, QoS)
- number of jobs (this user, other users)
- other jobs

Job placement:

- any free core on any compute node in the requested partition
- non-allocated memory \leq job mem req

Dependent jobs (“workflows”)

- job B can only run after job A

cores

```
$ sbatch jobA.sh
```

```
Submitted batch job 1234
```

```
$ sbatch --dependency=afterok:1234 jobB.sh
```

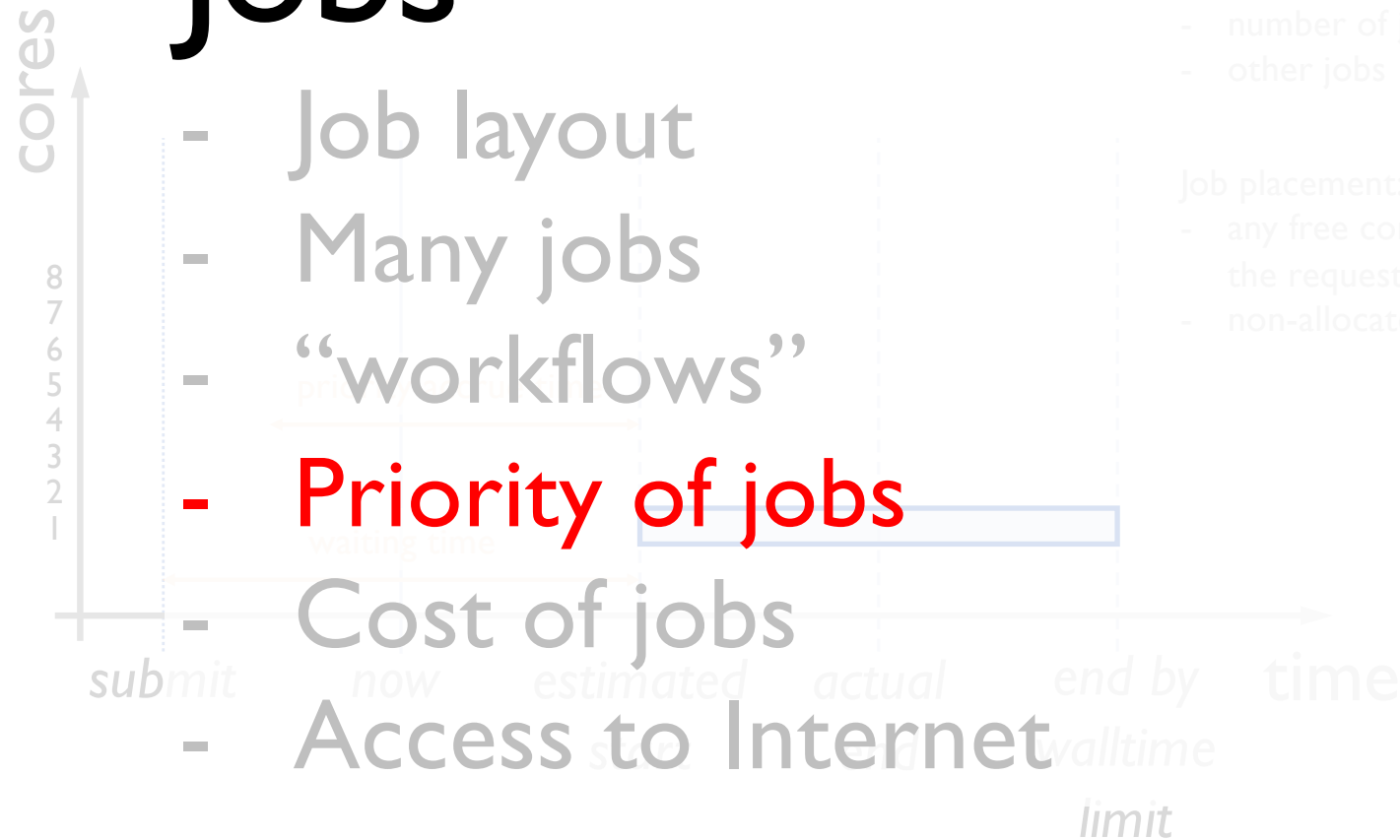
```
Submitted batch job 1240
```



```
$ squeue -u $USER
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (REASON)
2211325	normal	OpenFOAM	thomarob	PD	0:00	1	(Dependency)
2211324	normal	OpenFOAM	thomarob	R	0:16	1	c4-7

Jobs



Waiting time factors:

- job size (#cores, RAM, wall lim, QoS)
- number of jobs (this user, other users)
- other jobs

Job placement:

- any free core on any compute node in the requested partition
- non-allocated memory \leq job mem req

Priority of jobs

- means to distinguish jobs: age, QoS, ...
- optimization sched.: only top-prio jobs considered
- users cannot set a specific priority for a job
- List priorities of current jobs

```
$ sprio -S -y | head
```

JOBID	PARTITION	PRIORITY	SITE	AGE	QOS	
1629131	normal		29840	0	10080	19760
1629137	normal		29840	0	10080	19760

submit now estimated actual end by time
start end walltime
limit

Waiting time factors:

- job size (#cores, RAM, wall lim, QoS)
- number of jobs (this user, other users)

Priority of jobs - Example

```
$ sbatch big.sh; date; sprio -u $USER
```

```
$ sleep 120; date; sprio -u $USER
```

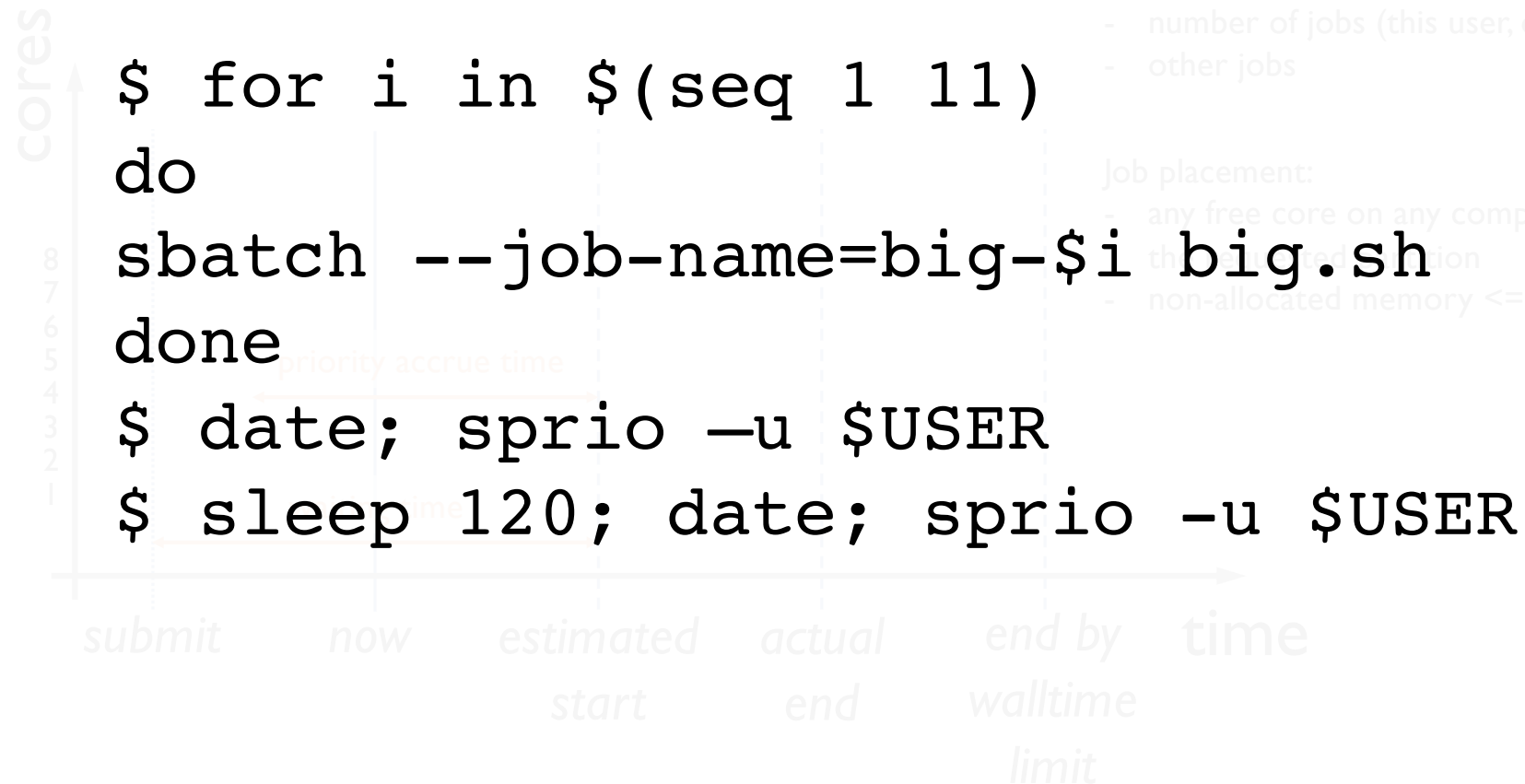
- every minute job ages by one
- max age 10080 (7 days)
- devel qos start priority is 120 higher
- only 10 jobs per user age

```
$ for i in $(seq 1 11); do sbatch --  
job-name=big-$i big.sh; done
```

```
$ date; sprio -u $USER
```

```
$ sleep 120; date; sprio -u $USER
```

Priority of jobs - Example



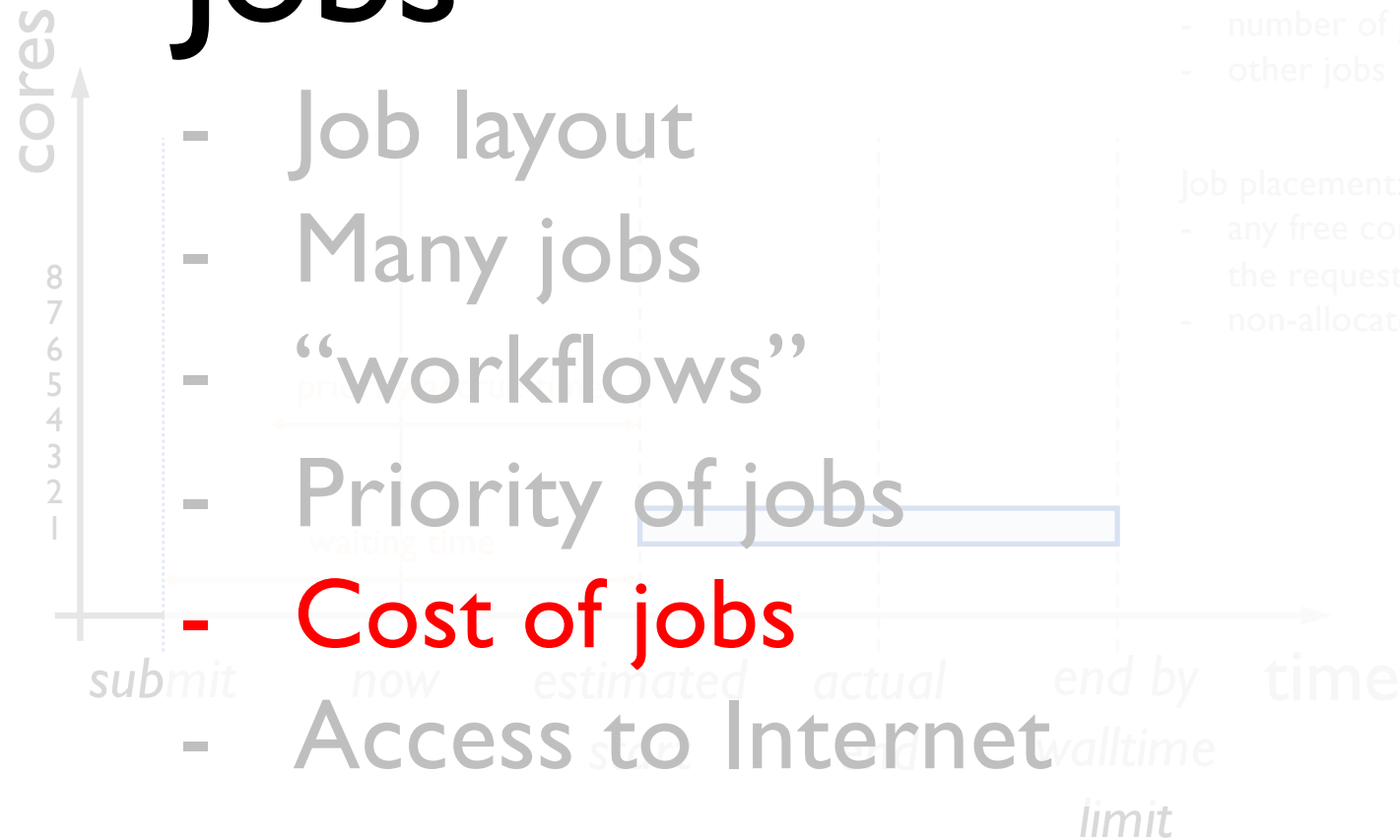
Waiting time factors:

- job size (#cores, RAM, wall lim, QoS)
- number of jobs (this user, other users)
- other jobs

Job placement:

- any free core on any compute node in the cluster
- non-allocated memory <= job mem req

Jobs



- Job layout
- Many jobs
- “workflows”
- Priority of jobs
- **Cost of jobs**
- Access to Internet

Waiting time factors:

- job size (#cores, RAM, wall lim, QoS)
- number of jobs (this user, other users)
- other jobs

Job placement:

- any free core on any compute node in the requested partition
- non-allocated memory \leq job mem req

Job accounting

- Every project (nnABCDk) has a quota (BU hours)
- BU = billing unit
- Job submission only successful when enough quota left
 - ➔ error label: AssocGrpBillingMinutes
- Jobs accounted for time actually used (not for time limit)
- Number of billing units = $f(\text{cores}, \text{RAM}, \text{GPUs})$
- Cost = **requested** #BU x actually used time
- Resources allocated to a job cannot be used by other jobs.

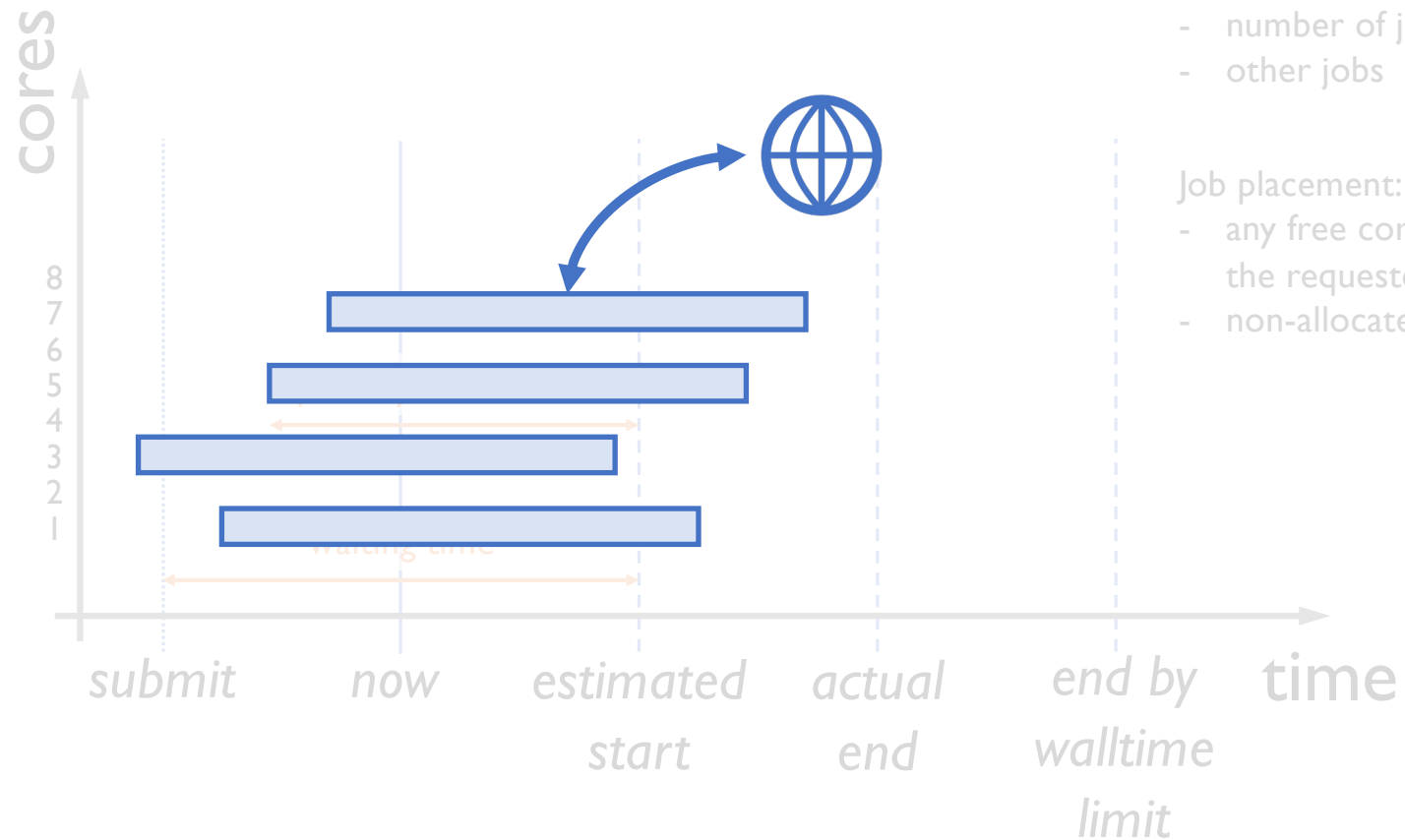
Job accounting – calculate cost

- #BU is maximum for BU for cores, memory and GPUs
- every core costs one BU (everywhere)
- every GPU costs six BU (Saga only)
- Memory costs proportionally to node “size”
- Higher costs on Saga’s bigmem nodes
- Request all mem incurs cost as if all CPU/GPU were used.
 - 377 GB on GPU nodes cost 24 BU
 - 186 GB on normal Saga nodes costs 40 BU
 - 377/3021 GB on bigmem Saga nodes costs 40/320 BU

Job accounting – Examples

- Saga: 40G, 4 cores, normal $\rightarrow \max(8.6, 4) = 8.6$
 - If your job can use more cores, you get 4 for free!
- Saga: 40G, 4 cores, bigmem $\rightarrow \max(4.2, 4) = 4.2$
 - Well balanced.
- Saga: 40G, 4 cores, 1 GPU $\rightarrow \max(2.5, 4, 6) = 6$
 - You could consider using more memory & cores.
- Fram: 1 node, normal $\rightarrow 32$ (node has 32 cores)
- Betzy: 4 nodes, normal $\rightarrow 512$ (node has 128 cores)

Access to Internet



Waiting time factors:

- job size (#cores, RAM, wall lim, QoS)
- number of jobs (this user, other users)
- other jobs

Job placement:

- any free core on any compute node in the requested partition
- non-allocated memory \leq job mem req

Access to Internet

- From jobs running on compute nodes
- GitHub (`git clone`)
- pages which contain data (`wget`, `curl`)
- Configured on Saga & Fram

```
$ srun --ntasks=1 --mem-per-cpu=4G --time=0:30:0 --  
account=nn9989k --pty bash -i  
...  
git clone https://github.com/EESSI/docs.git  
wget https://eessi-hpc.org  
curl -I https://www.sigma2.no
```

Waiting time factors:

- job size (#cores, RAM, wall lim, QoS)
- number of jobs (this user, other users)

Job placement:

- a free core on any compute node in the requested partition
- non-allocated memory \leq job mem req