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

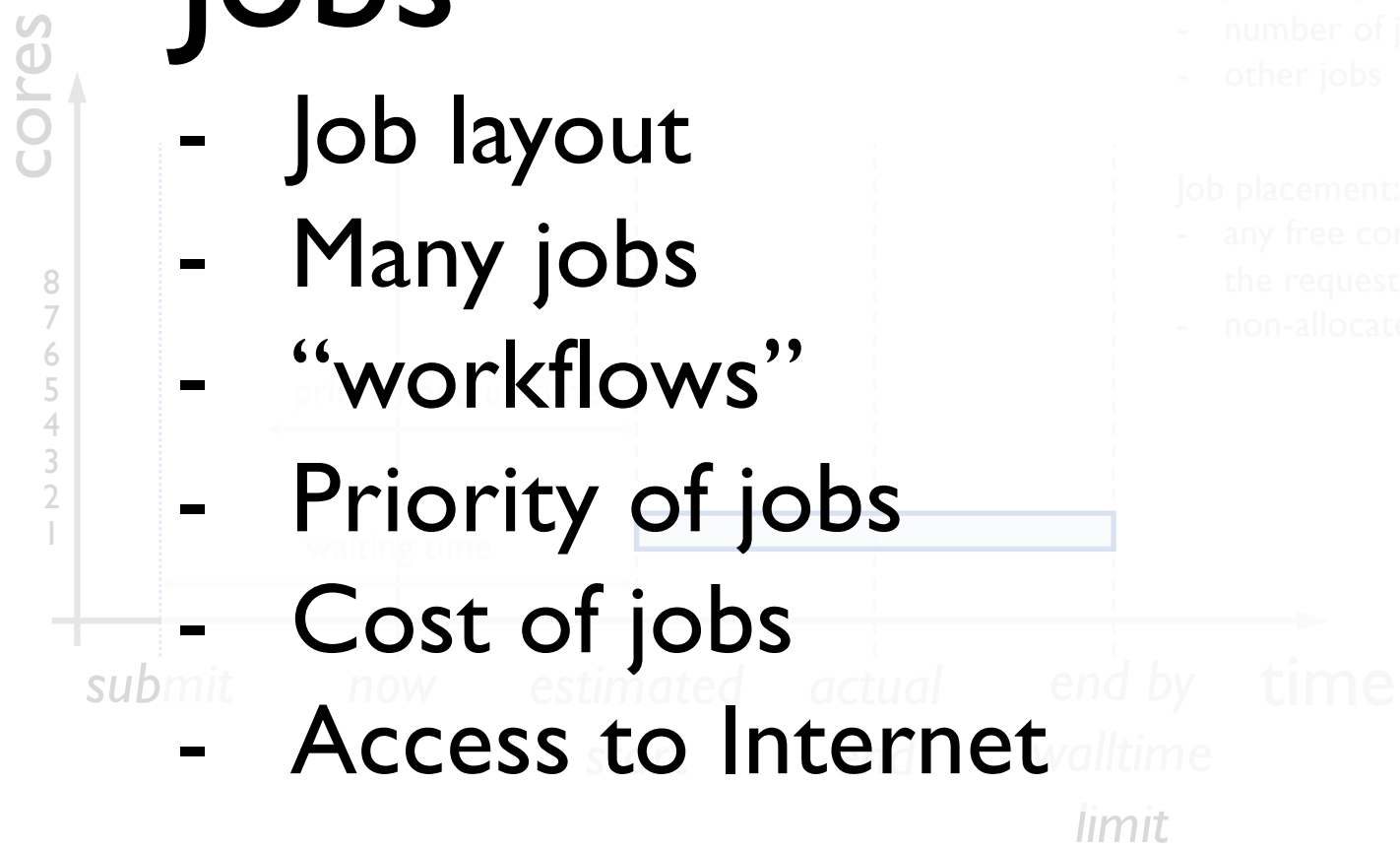
Me

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

EESSI
EUROPEAN ENVIRONMENT FOR SCIENTIFIC SOFTWARE INSTALLATIONS

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

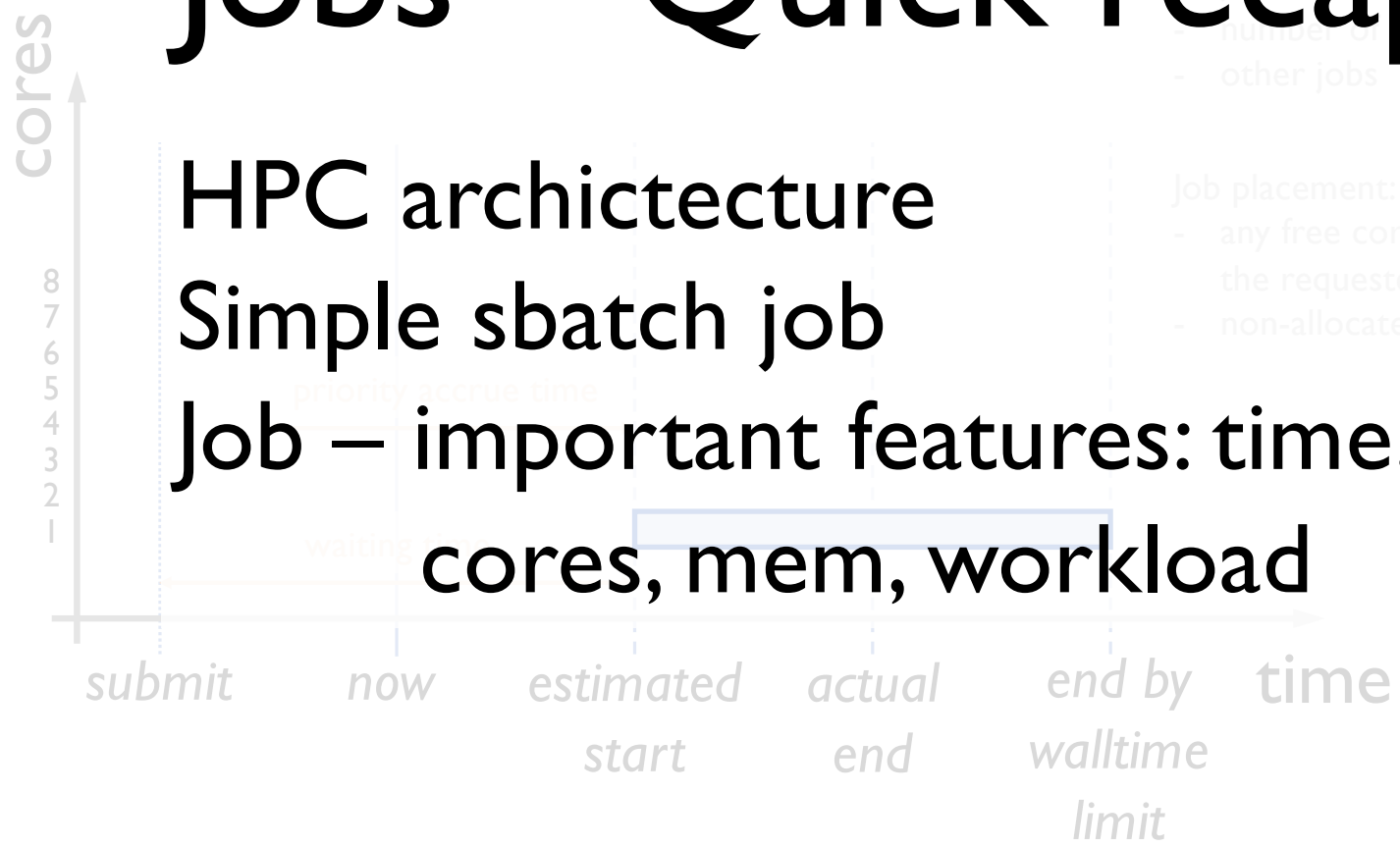
Job placement:
- any free core on any compute node in the requested partition
- non-allocated memory <= job mem req

Jobs – Quick recap

HPC archictecture

Simple sbatch job

Job – important features: time, cores, mem, workload

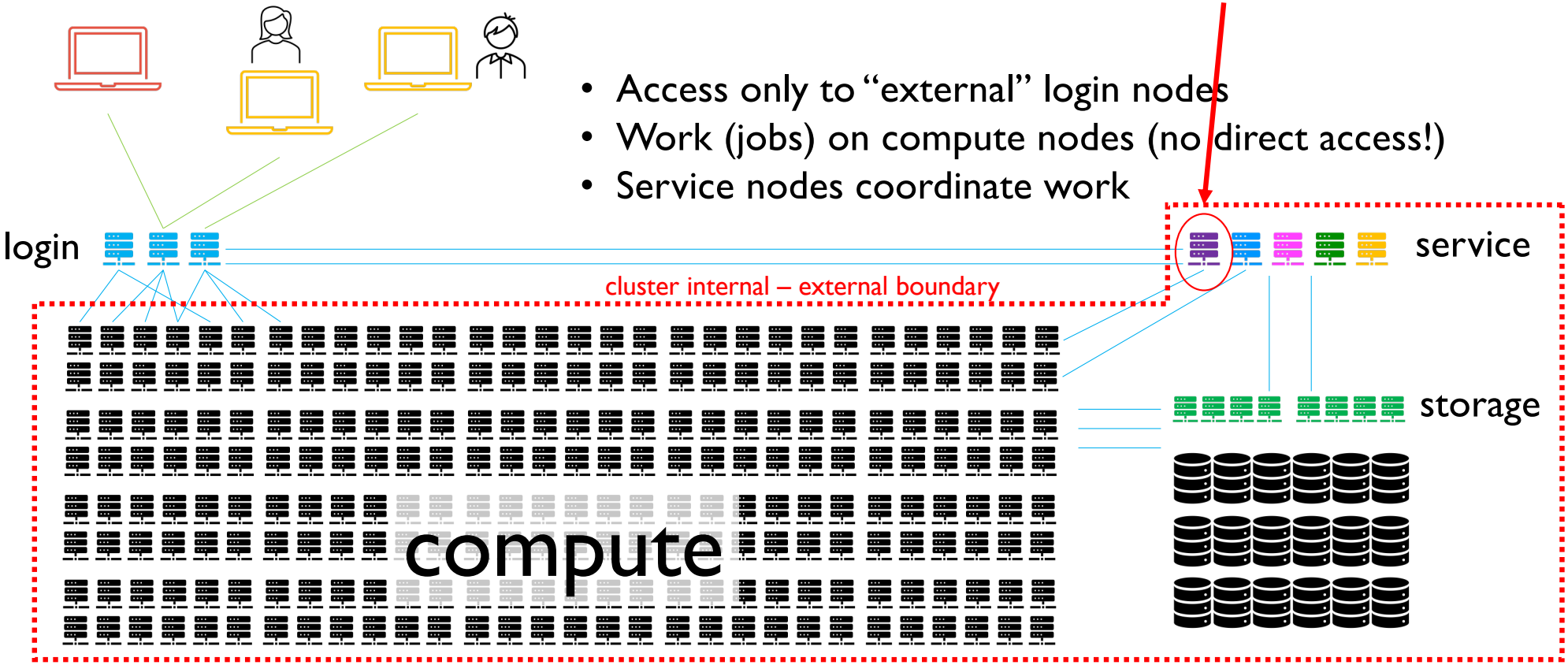


Waiting time factors:
- size (cpu, RAM, wall lim, QoS)
- number of jobs (this user, other users)
- other jobs

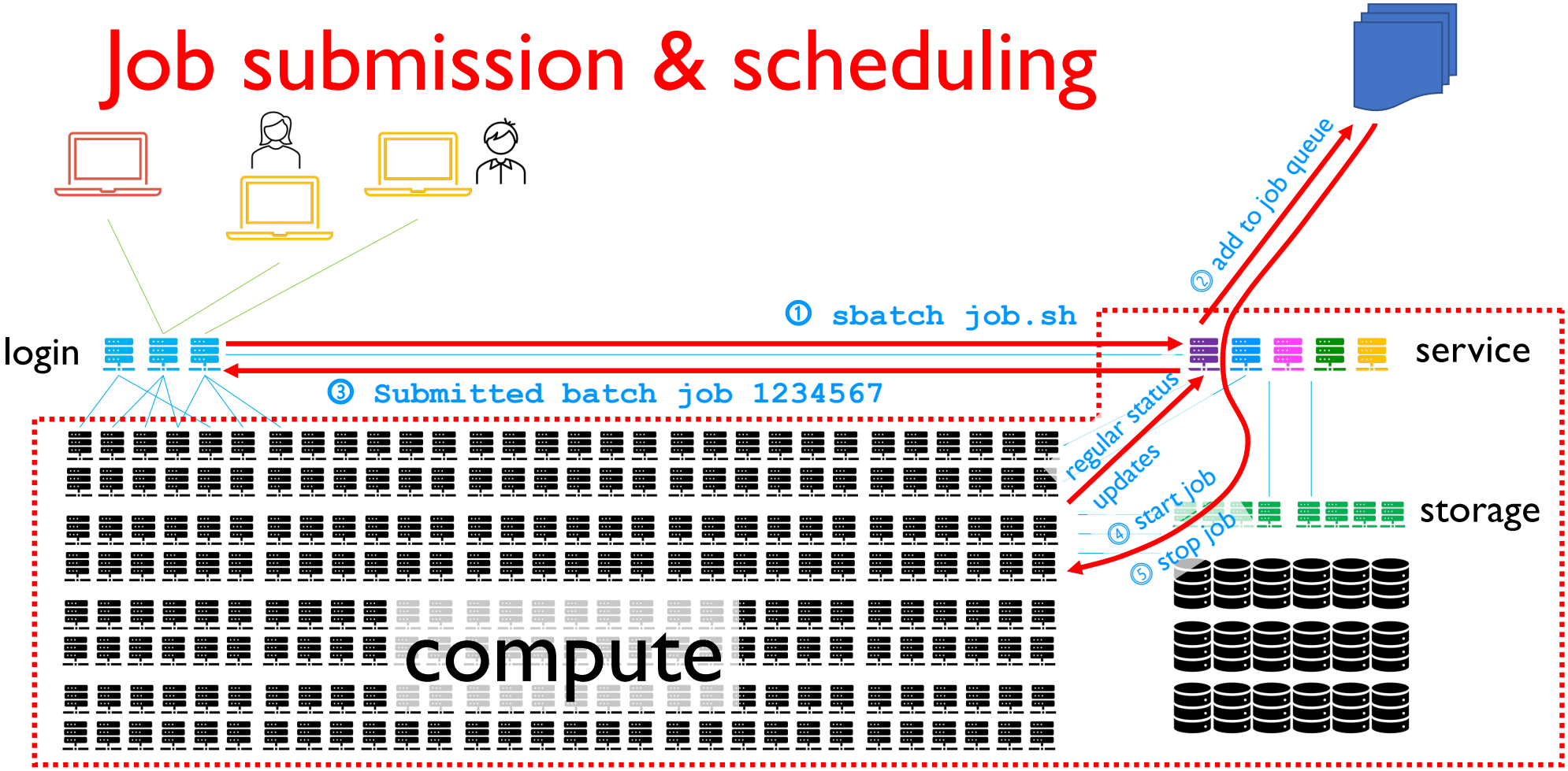
Job placement:
- any free core on any compute node in the requested partition
- non-allocated memory <= job mem req

HPC architecture

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



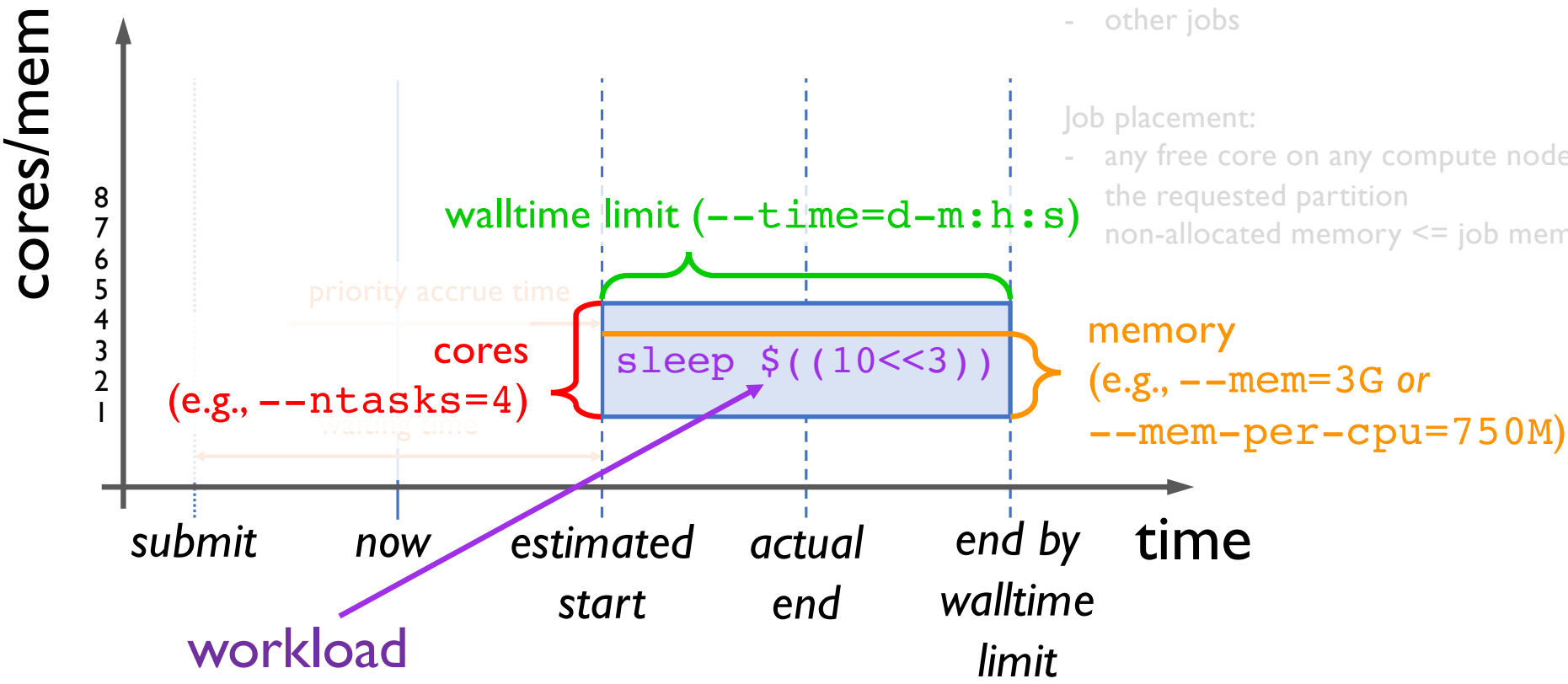
Job submission & scheduling



Jobs – important features

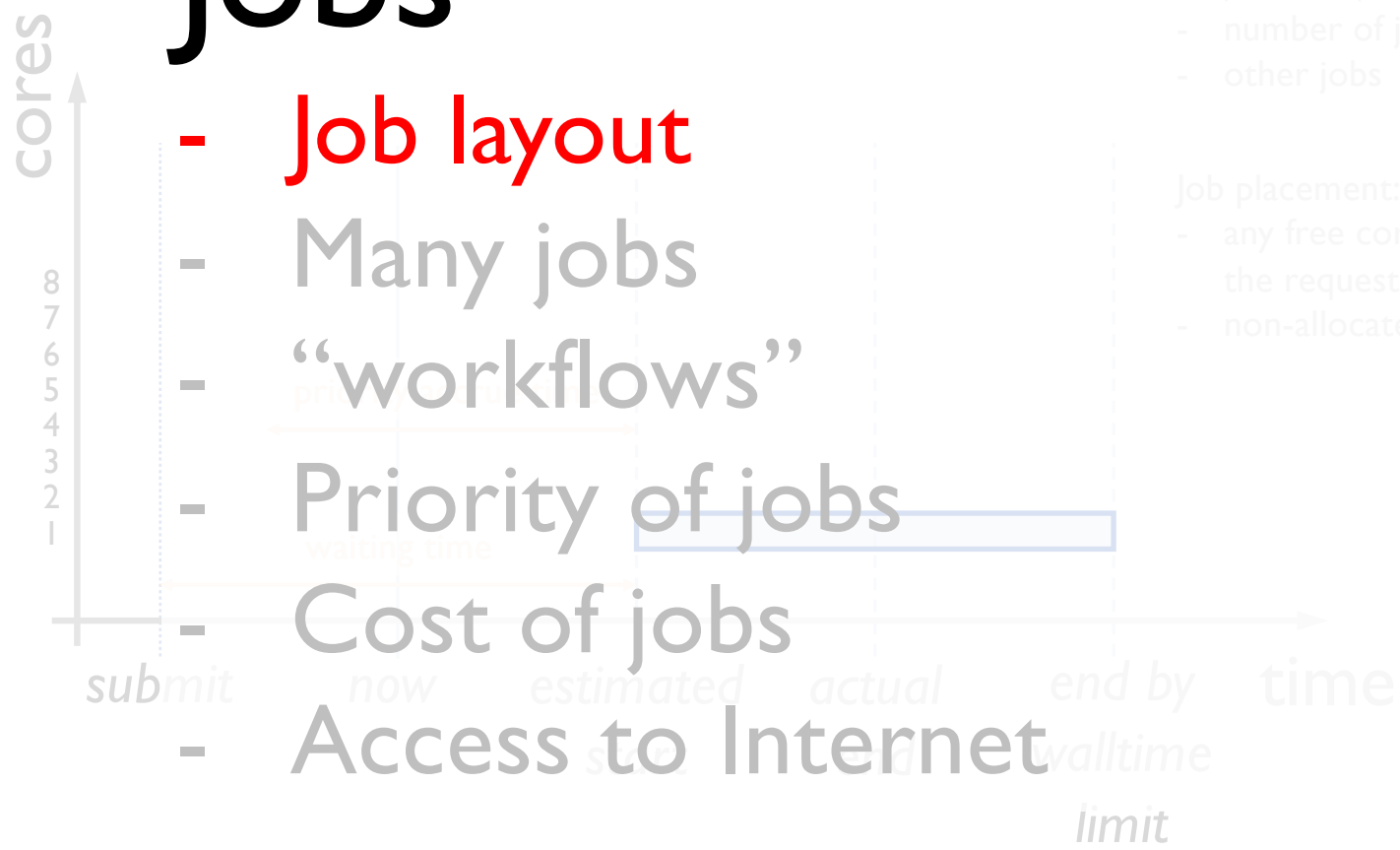
Waiting time factors:
- job size (#cores, RAM, wall lim, QoS)
- number of jobs (this user, other users)
- other jobs

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

Job layouts

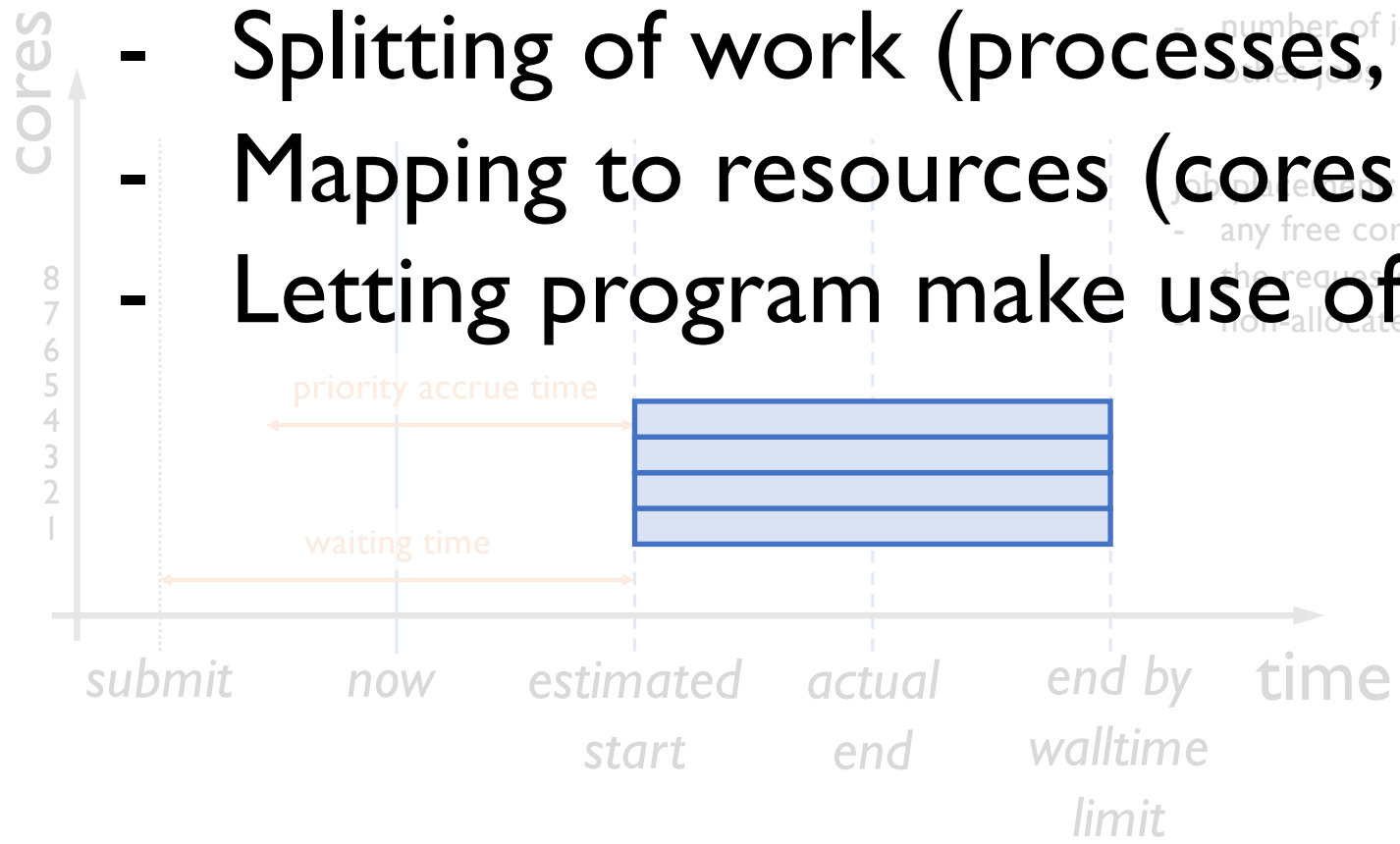
- Splitting of work (processes, threads)
- Mapping to resources (cores, nodes)
- Letting program make use of layout

Waiting time factors:

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

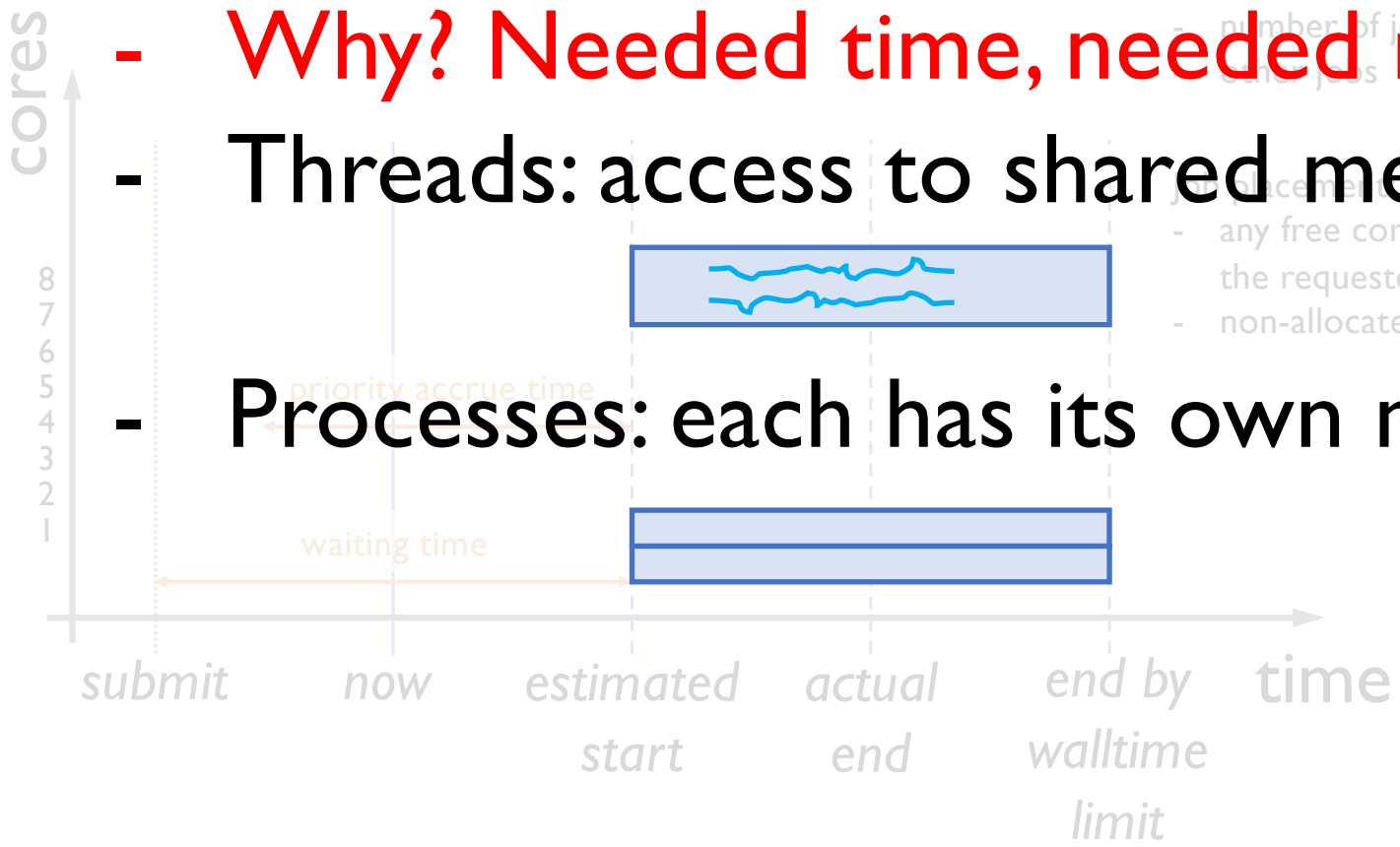
Choice criteria:

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



Job layouts – splitting of work

- Why? Needed time, needed memory
- Threads: access to shared memory
- Processes: each has its own memory

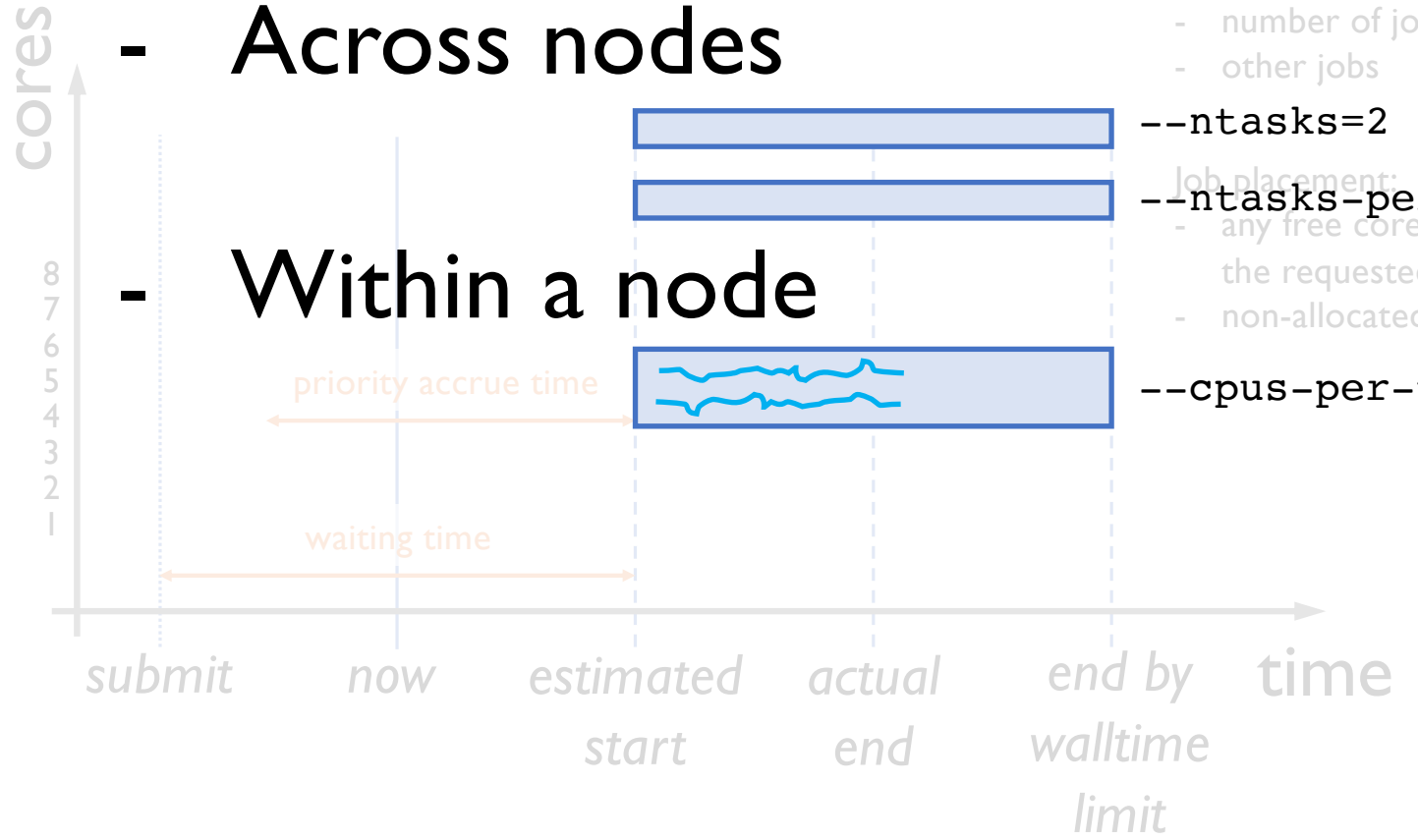


Waiting time factors:
- job size (#cores, RAM, wall lim, QoS)
- number of jobs (this user, other users)
- other jobs

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

Job layouts – Mapping to resources

- Across nodes
- Within a node




```
--ntasks=2
--ntasks-per-node=1 --nodes=2
--cpus-per-task=2 --nodes=1
```

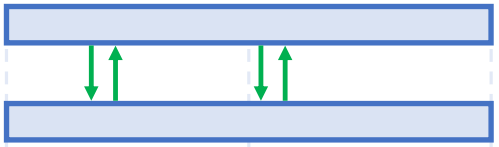
- job size (#cores, RAM, wall lim, QoS)
- number of jobs (this user, other users)
- other jobs

Job placement:

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

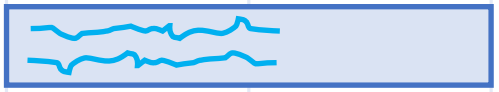
Job layouts – Let a program use layout

- Message Passing Interface (Across nodes)



```
srun program args
mpirun program args*
```

- OpenMP (Within a node)



```
export OMP_NUM_THREADS=2
program args
```

(*) 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-tasks=4
...
program args
...
```

Job layouts – Examples

- Job uses 4 processes (tasks) *anywhere*

```
...
#SBATCH --ntasks=4
...
srun program args
...
```

Job layouts – Examples

- 2 nodes, 4 processes (tasks)

```
...
#SBATCH --nodes=2
#SBATCH --ntasks=4
...
srun program args
...
```

Job layouts – Examples

- 2 nodes, 2 processes (tasks) on each

```
...
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=2
...
srun program args
...
```

Job layouts – Examples

- 4 processes (tasks), 2 on each node

```
...
#SBATCH --ntasks=4
#SBATCH --ntasks-per-node=2
...
srun program args
...
```

Job layouts – What to use & when?

- Computing effort
  - How long does it run? test small
- Problem size
  - How much mem test small, read doc
- Program ability
  - 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

- 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 <= 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

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

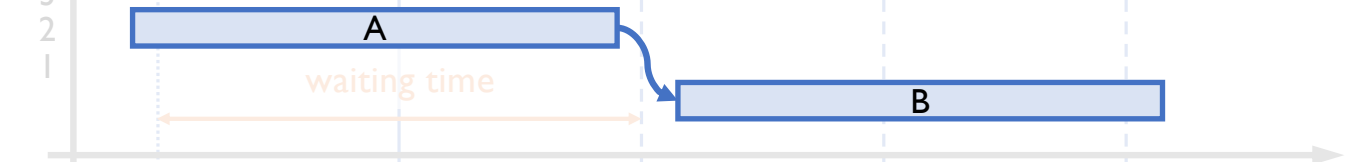
Job placement:

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

Dependent jobs ("workflows")

- job B can only run after job A

```
$ 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
```

https://documentation.sigma2.no/jobs/guides/job_dependencies.html

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 <= 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
```

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

```
$ 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
```

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 <= 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(cores, RAM, 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

https://documentation.sigma2.no/jobs/projects_accounting.html

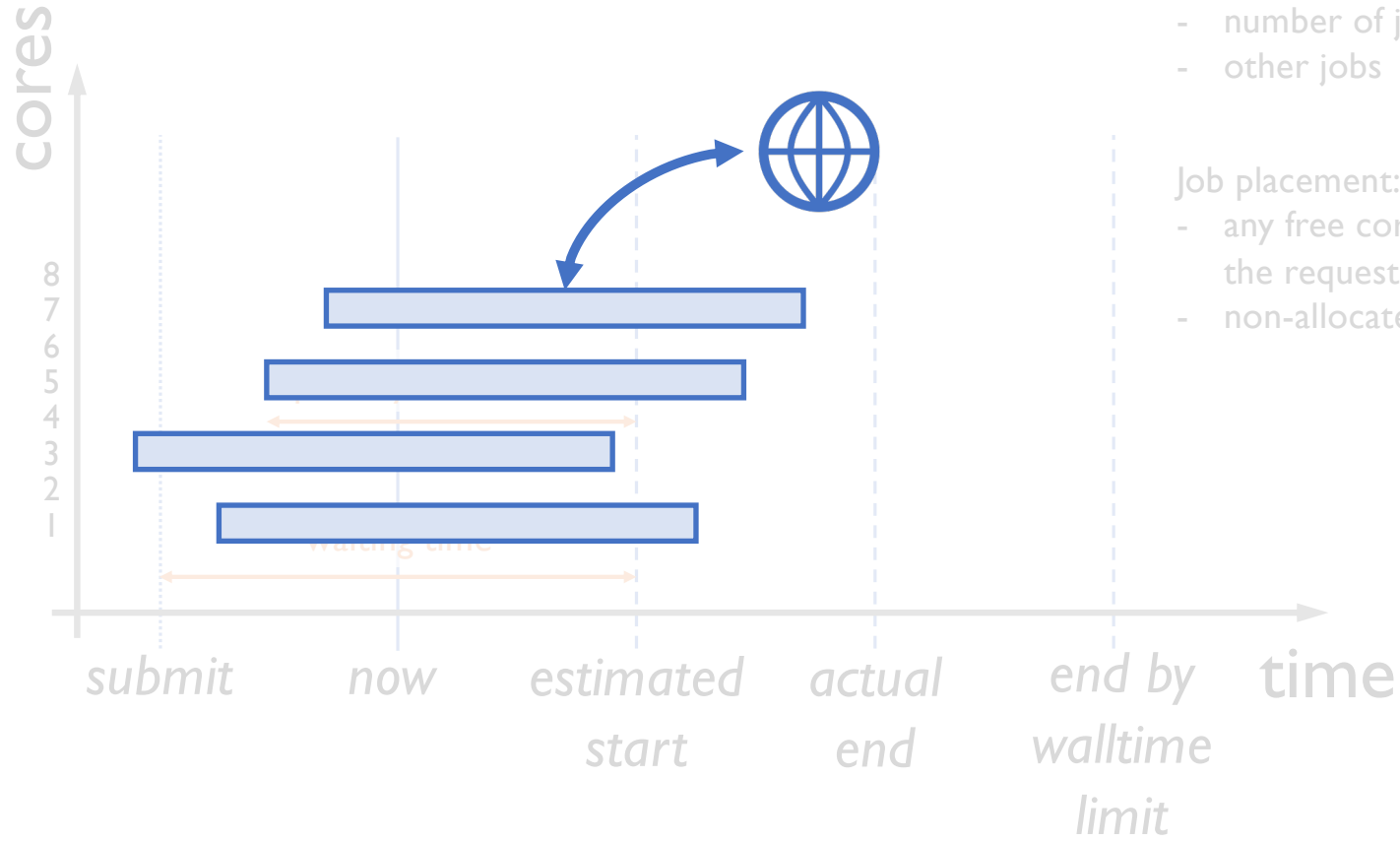
Job accounting – Examples

- Saga: 40G, 4 cores, normal ➔ max(8.6,4) = 8.6
  - If your job can use more cores, you get 4 for free!
- Saga: 40G, 4 cores, bigmem ➔ max(4.2,4) = 4.2
  - Well balanced.
- Saga: 40G, 4 cores, 1 GPU ➔ max(2.5,4,6) = 6
  - You could consider using more memory & cores.

- Fram: 1 node, normal ➔ 32 (node has 32 cores)
- Betzy: 4 nodes, normal ➔ 512 (node has 128 cores)

https://documentation.sigma2.no/jobs/projects_accounting.html

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 <= 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
```

Exercise time

- We use the EESSI pilot software stack running GROMACS.
- Goal is to run GROMACS with a certain payload (number of steps: 20000).
- Example scripts for Saga and Fram as well as instructions available on GitHub
- Overall approach: start small, experiment, scale up
- Start on Saga. When you reached 4-8 ntasks, move to Fram.

```
git clone https://github.com/trz42/HPC-NTK-user-course-2021.git
```