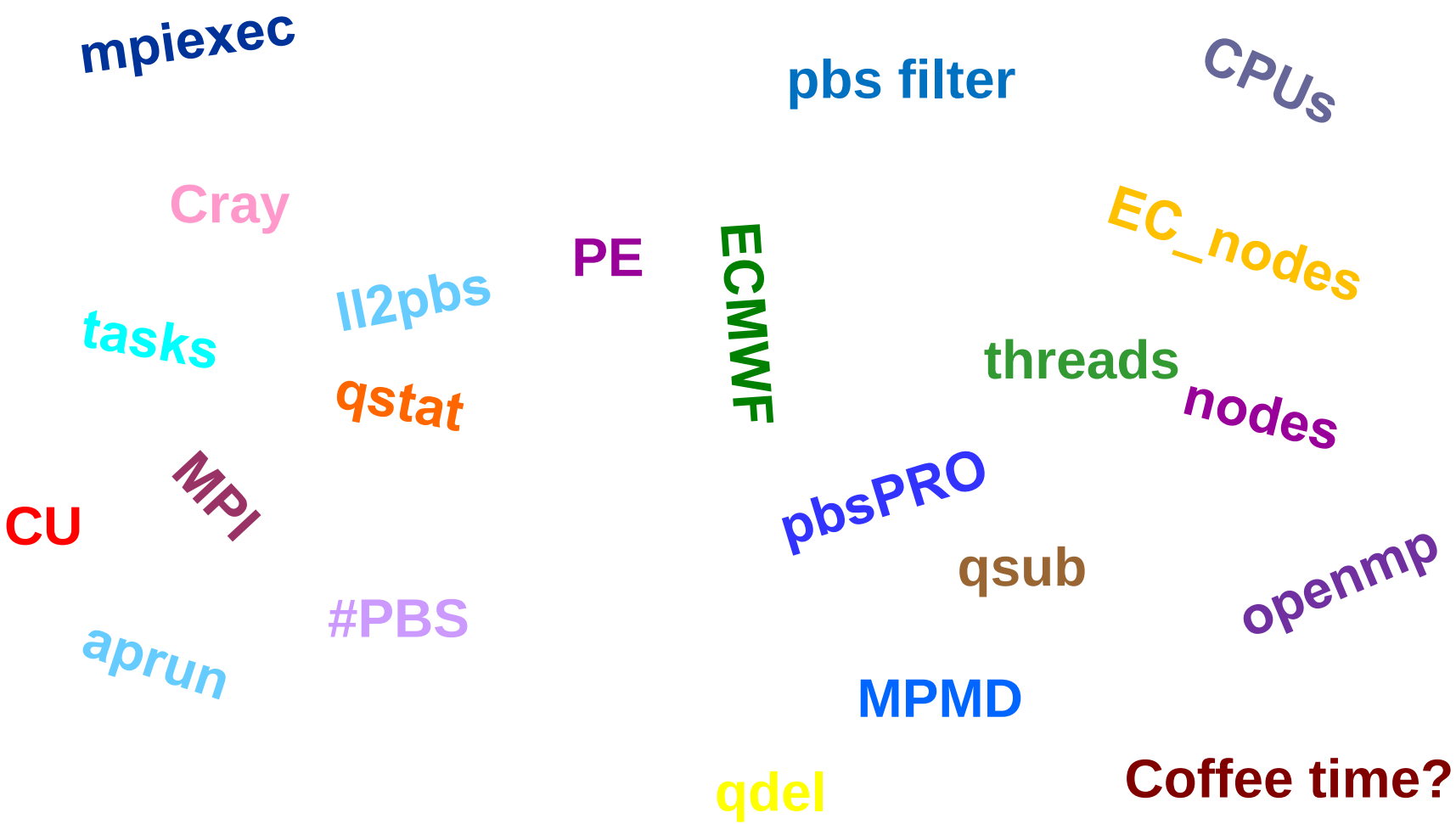
PBS(pro) at ECMWF

Dominique Lucas
User Support

Agenda



Current (Cray XC30) vs. Old (IBM Power7)

	Old	Current
Sustained performance	~70 teraflops	~ 200 teraflops
Peak performance	~1500 teraflops	~3600 teraflops
Compute clusters	2	2
Each compute cluster		
Compute nodes	739	~3,500
Compute cores	23,648	~84,000
Total memory (TB)	46	~210
Pre-/post-processing nodes	20	100
Operating System	AIX 7.1	SUSE Linux/CLE
Scheduler	IBM LoadLeveler	Altair PBSpro/ALPS
Interconnect	IBM HFI	Cray Aries

Different User node types for batch work on Cray

- MOM
  - Where the parallel job scripts run
  - Need to minimise the serial content of such scripts
- Pre and Post Processing (PPN)
  - Used for serial jobs
  - Used for small parallel jobs requiring less than half a node
- Extreme Scalability Mode (ESM) (Compute Nodes) (CN)
  - Where the multi node parallel executables run
  - Accessed via the command ‘aprun’ (equivalent of mpirun, mpiexec, …) from the MOM node

- IBM had just one type of node.

PBSpro nodes (commands run on cca)

- pbsnodes –ar | less

```
ccamom05 # this is a MOM node
    ntype = PBS
    jobs = 1036567.ccapar/0, 1036565.ccapar/0, 1036364.ccapar/0. …
    resources_available.EC_accept_from_queue = np,dp,tp
    resources_available.vntype = cray_login
ccappn007 # this is a pre/post processing node (PPN)
    jobs = 1036445.ccapar/1, 1035396.ccapar/2, 1036450.ccapar/3, ...
    resources_available.EC_accept_from_queue = os,ts,ns,of,tf,nf,df,ds
    resources_available.vntype = cray_postproc
cca_2140_0  # this is (half) a Compute node (CN)
    resources_available.EC_accept_from_queue = np,tp,op, dp
    resources_available.vntype = cray_compute
```

- apstat

```
Compute node summary
   arch config     up   resv    use  avail   down
    XT  3400   3395 3317  2964  78      5
```


PBSpro basics

- Directives in batch jobs start with '#PBS'
- Main User commands:

User Commands	SLURM (ecgate)	PBSpro
Job submission	sbatch <script>	qsub [options] <script>
Job cancel	scancel <job_id>	qdel <job_id>
Job status	squeue	qstat [job_id]
Queue list	sqos*	qstat –Q [-f] [queue]
		qscan*
Check job output		qcat*

(*) ECMWF local commands.

Batch queues on ECMWF Cray HPCs

User Queue Name	Suitable for	Target nodes	Number of processes min/max	Shared / not shared	Processes per node available for user jobs
ns	serial	PPN	1/1	shared	48
nf	fractional	PPN	2/24	shared	48
np	parallel	MOM+CN	1/48	not shared	48

- Similar queues for time critical (option 2) work: ts, tf, tp.
- Debug queues are also available: ds, df and dp.
- ‘ qstat –Q –f <queue_name> ’ gives full details on specified queue

Some queue limits

- Normal queues:
  - User jobs run limit: 20 jobs per queue
  - Time limit: 2 days
  - Memory limit:
    - In queue np: all the memory available (~60GB/node)
    - In queues ns and nf, no memory limits enforced yet. Try to specify the limit you need.
- Time critical queues:
  - Varying user jobs run limits
  - Shorter time limits

PBSpro environment variables

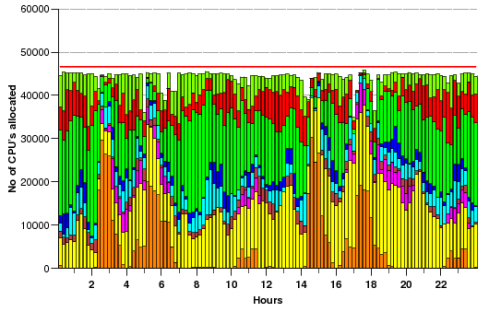
- A number of environment variables are set by PBSpro for all jobs:

```
PBS_ENVIRONMENT=PBS_BATCH
PBS_JOBCOOKIE=000000004D0B31AC000000007AE6B946
PBS_JOBDIR=/home/ectrain/trcray0
PBS_JOBID=124948.sdb
PBS_JOBNAME=save_results
…
```

… plus many, many more by cray-pe et al., and by the ECMWF PBS Hook …

see later …

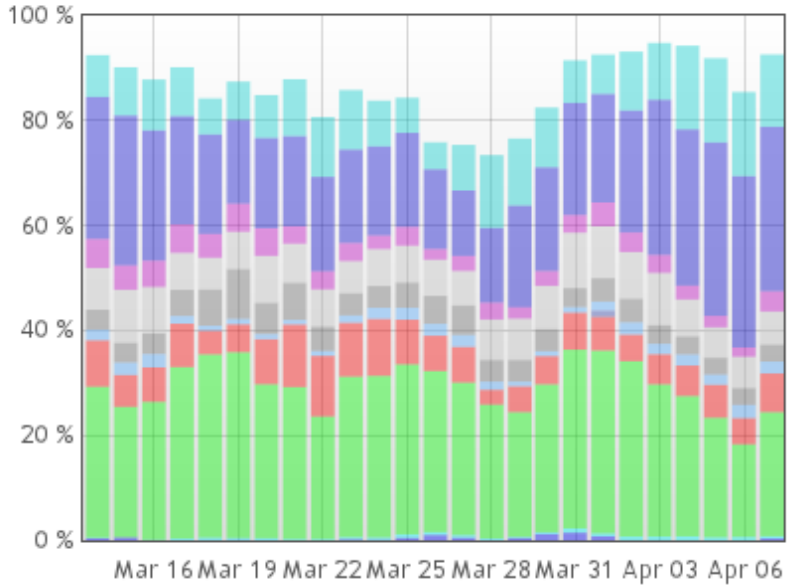
ECMWF operational work



- Tight schedule, huge resources needed at short notice
- Three options:
  - Exclusive HPC resources for operational work
  - Sharing of HPC resources with default scheduling of batch work
  - Sharing of resources with enhanced scheduling of batch work

Figures:

- On the right, cca cluster node occupancy for 1 month in 2015: ~85%. In green, MS work.
- Top right, IBM (previous HPC) occupancy: ~95%



ECMWF's enhanced scheduling

- Usage of PBSpro job reservation system.
- For this, we need a precise description of the jobs, including runtime and node requirements.
- The ECMWF PBS Hook will set the node requirements and assign an estimated runtime for each job, to guarantee an optimal usage of the resources and a timely delivery of the operational data.
- A jobs runtime database keeps the wall clock time of the last 20 runs for not more than 30 days.
- To benefit from database and scheduling:
  - Keep job name and output file name identical for the same jobs.

PBSpro User perspective – ECMWF Hook

- Users want something ‘simple’ to write their jobs
- The node requirements and job geometry for a job in PBSpro are specified with a ‘select’ statement (#PBS –l select=..’).
- This statement is quite complex to use and doesn’t cover all the requirements needed for the ECMWF enhanced scheduling.
- Another complication with PBSpro is that the user will have to redefine the geometry of his/her run in the script (with ‘aprun’).
- At ECMWF, we have therefore decided to customise the PBSpro environment.
  - Users cannot write their own ‘select’ statement.
  - ECMWF PBS directives are available to define the job geometry.
  - Environmental variables defining the job geometry are available in the job script.

Reminder: some terminology (Cray vs. IBM and others)

- 1 PE (processing element) = 1 MPI rank = 1 (MPI) task
- 1 Computational Unit (CU) = 1 core = 1 physical CPU
- 1 CPU = 1 logical CPU (hyperthreading, Simultaneous Multi Threading SMT)
- 1 node = 2 NUMA nodes

ECMWF pbsPRO directives and variables

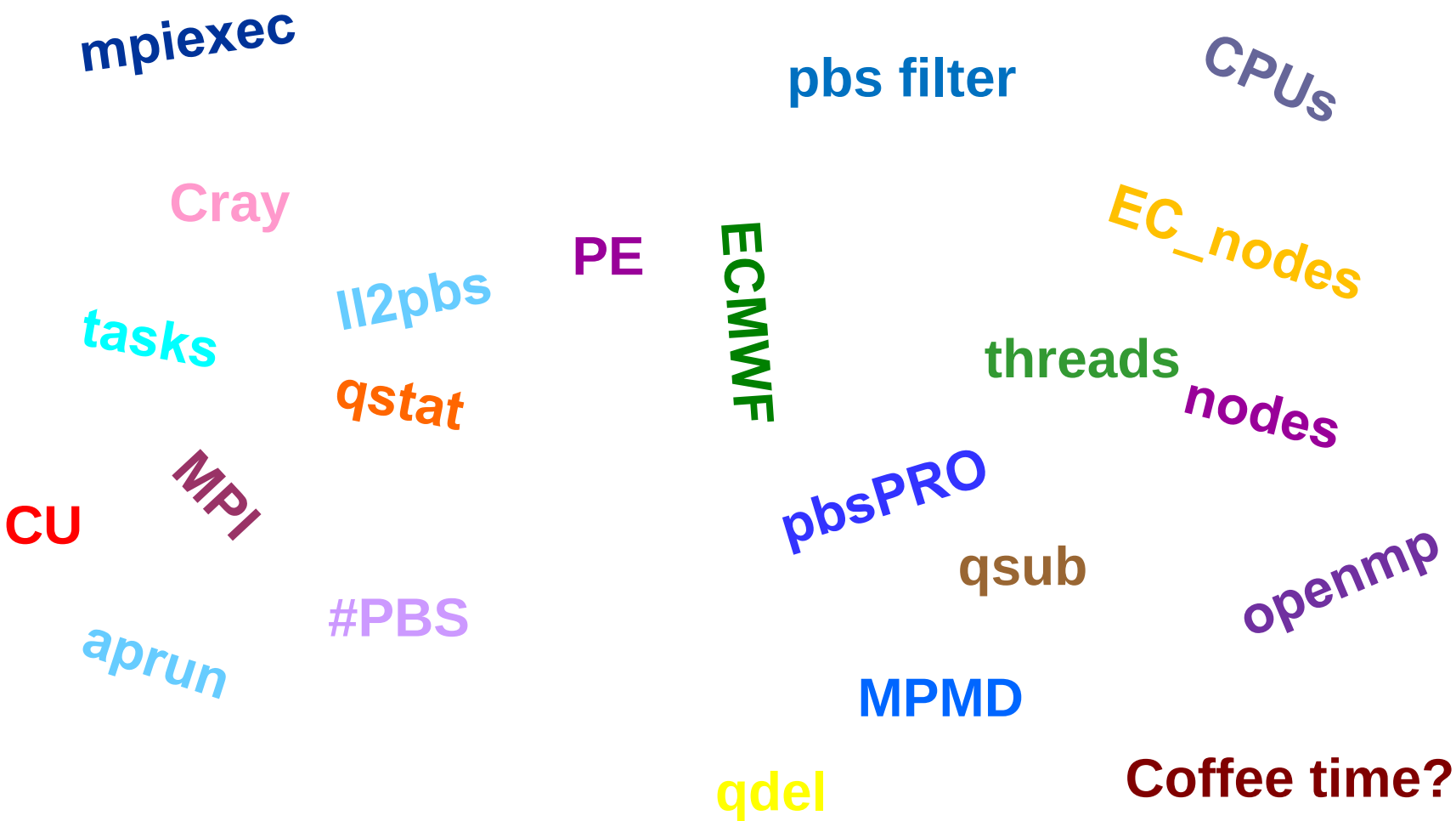
Directive name	Description	Default
EC_total_tasks	Total number of tasks requested	1
EC_threads_per_task	Number of threads per task	1
EC_tasks_per_node	Maximum number of tasks per node	24/48
EC_nodes	Number of nodes requested	1
EC_memory_per_task	Amount of memory required per task	100MB
EC_hyperthreads	Number of CPUs per CU	1

- These directives are given like '#PBS –l <EC_name>=<value>'.
- These directives can be passed as options to qsub, e.g.:

  qsub –l EC_total_tasks=48 job.cmd

- The pbs Hook will define variables in the job with the same as the names for the 'EC_' directives.

Half time … questions?



ECMWF job example 1 – DIY (do it yourself) option

➢ cat HelloMPIandOpenMP.cmd

```
..
#PBS -N HelloMPI_OMP
#PBS -q np
#PBS -l EC_nodes=3
…
export OMP_NUM_THREADS=6
aprun -n 24 -d 6 -j 2 HelloMPI_OMP
```

➢ qsub HelloMPIandOpenMP.cmd

124950.ccbpar

➢ qstat –f 124950.ccbpar

```
…
 Resource_List.select=1:vntype=cray_login:EC_accept_from_queue=dp:ncpus=0:
mem=300MB+3:vntype=cray_compute:EC_accept_from_queue=dp:mem=60GB
```


ECMWF job example 2 – Flexible option

➢ cat HelloMPIandOpenMP.cmd

```
..
#PBS -N HelloMPI_OMP
#PBS -q np
#PBS -l EC_total_tasks=30
#PBS -l EC_threads_per_task=3
#PBS -l EC_memory_per_task=3GB
#PBS -l EC_hyperthreads=2
…
export OMP_NUM_THREADS=$EC_threads_per_task
aprun -N $EC_tasks_per_node -n $EC_total_tasks \
-d $EC_threads_per_task -j $EC_hyperthreads ./HelloMPI_OMP
…
```


ECMWF job example 3 – MPMD programs, e.g. coupled runs

```
➢ cat MPMD.cmd
..
#PBS -N HelloMPMD
#PBS -q np
#PBS -l EC_total_tasks=24:6
#PBS -l EC_threads_per_task=1:1
#PBS -l EC_hyperthreads=1
…
IFS=':'
set –A tasks_per_node $EC_tasks_per_node
set –A total_tasks $EC_total_tasks
aprun -n ${total_tasks[0]} –N ${tasks_per_node[0]} ./model_atm : \
        -n ${total_tasks[1]} –N ${tasks_per_node[1]} ./model_ocean
…
```

With this method, one cannot run two different executables on the same node, therefore wasting CPU resources.

ECMWF job example 4 – MPMD programs, improved recipe

➢ cat MPMD_turbo.cmd

```
..
#PBS -N HelloMPMD
#PBS -q np
#PBS -l EC_nodes=1

cat>wrapper.sh<<\eof
#!/bin/ksh
rank=$EC_FARM_ID
export OMP_NUM_THREADS=1
if [ $rank -lt 17 ]; then
   exec ./atm>atm.out.$rank 2>&1
else
   exec ./ocean>ocean.out.$rank 2>&1
fi
eof

export PMI_NO_FORK=1
# this variable is essential
aprun –n24 –N24 -j1 ./wrapper.sh



# $EC_FARM_ID is defined for
# you fir each entity of the
# wrapper.sh script running.
```

https://cug.org/proceedings/cug2014_proceedings/includes/files/pap136.pdf for more details.

No waste of CPU resources!

ECMWF job example 5 – fractional job

➢ cat small_job.cmd

```
..
#PBS -N HelloMPI_S
#PBS -q nf
#PBS -l EC_total_tasks=12
#PBS -l EC_hyperthreads=2
…
module load cray-snplauncher
mpiexec -n $EC_total_tasks ./HelloMPI_S
…
```


Additional ECMWF PBSpro directives

- If your job uses MARS, you can add
  - #PBS –l EC_mars=1
- If your job uses ECFS, you can add
  - #PBS –l EC_ecfs=1

The two above directives will help us to schedule the work better.

- Final remark on ECMWF PBS Hook.
  - The PBS Hook has covered all the user requirements so far.
  - It can and will be adapted, if needed, for any new requirement.
  - The Hook may correct invalid job geometries.
  - Verbose output from the Hook is added in the job output file.
  - More information on the ECMWF pbsPRO Hook under:

https://software.ecmwf.int/wiki/display/UDOC/Batch+environment%3A++PBS

Multi step jobs (IBM LoadLeveler facility)

- Typical 3-steps job, with dependencies:

fetch data	- serial
model run	- parallel
post processing	- serial

- This facility is not available under PBSpro in the same (simple) format.
- Alternatives are to:
  - use the ‘qsub –W depend’ option
  - use ‘qsub –W block=true’ option
  - use a scheduling system, e.g. ECMWF’s ecFlow (or SMS) software.

Various

- ksh and bash shells are supported with PBS
- The ECMWF command ‘eoj’ (end of job) is available and included at the end of each job output file.
  - The resource usage reported in ‘eoj’ is only correct for queues ‘ns’ and ‘nf’. For ‘np’, only resources spent on the MOM node are reported in ‘eoj’. See aprun final report line for the usage of parallel resources.
- Job accounting is running for PBS:
  - All jobs will be charged for elapsed time times the number of CU utilised
- Try out different geometries for your job and choose the optimal one.

Various (cont)

- ‘Interactive batch’ sessions:

```
➢ qsub -I -q np -l EC_nodes=1-X
qsub: waiting for job 9848342.ccbpar to start …
qsub: job 9848342.ccbpar ready
...
ccbmom06:> aprun …
```

- From remote (non ECMWF) systems, via ECaccess, to start GUI applications:

```
➢ echo $DISPLAY $X11PROTOCOL $X11COOKIE
136.156.66.24:0.0 MIT-MAGIC-COOKIE-1 3ce0e9a973020f4fe559dd7d108c5195
➢ qsub -I -q np -l EC_nodes=1 -X
qsub: waiting for job 9848342.ccbpar to start …
qsub: job 9848342.ccbpar ready
...
ccbmom06:> xauth add 136.156.66.24:0.0 MIT-MAGIC-COOKIE-1 3ce0e9a973020f4fe559dd7d108c5195
ccbmom06:> xclock
```


Cray Phase I (XC30) upgrade to Phase II (XC40)

	Phase I	Phase II
Sustained Performance (teraflops)	200	320
Peak performance (teraflops)	3,593	~8,500
Processor technology	Intel Ivy Bridge	Intel Broadwell
Parallel application nodes	3,400	3,513
Cores per node	24	36
Memory per node (GiB)	64 (1866 MHz DDR3)	128 (2400 MHz DDR4)
External login nodes	2 x Ivy Bridge	2 x Ivy Bridge, 1 x Haswell
Clock frequency (GHz)	2.7	2.1
Storage capacity (petabytes)	15	20
Floating Point Instruction set	AVX	AVX2
Default compiler	Cray 8.2.7	Cray 8.4.x

Batch queues on Crays XC40

User Queue Name	Suitable for	Target nodes	Number of processes min/max	Shared / not shared	Processes per node available for user jobs
ns	serial	PPN	1/1	shared	72
nf	fractional	PPN	2/36	shared	72
np	parallel	MOM+CN	1/72	not shared	72

- You may need to adapt the 'EC_' directives to the new node:

```
#PBS –l EC_total_tasks=48
#PBS –l EC_threads_per_task=3
```

runs on 3 nodes on XC30 and will run on 2 nodes on XC40. No change needed.

```
#PBS –l EC_total_tasks=48
```

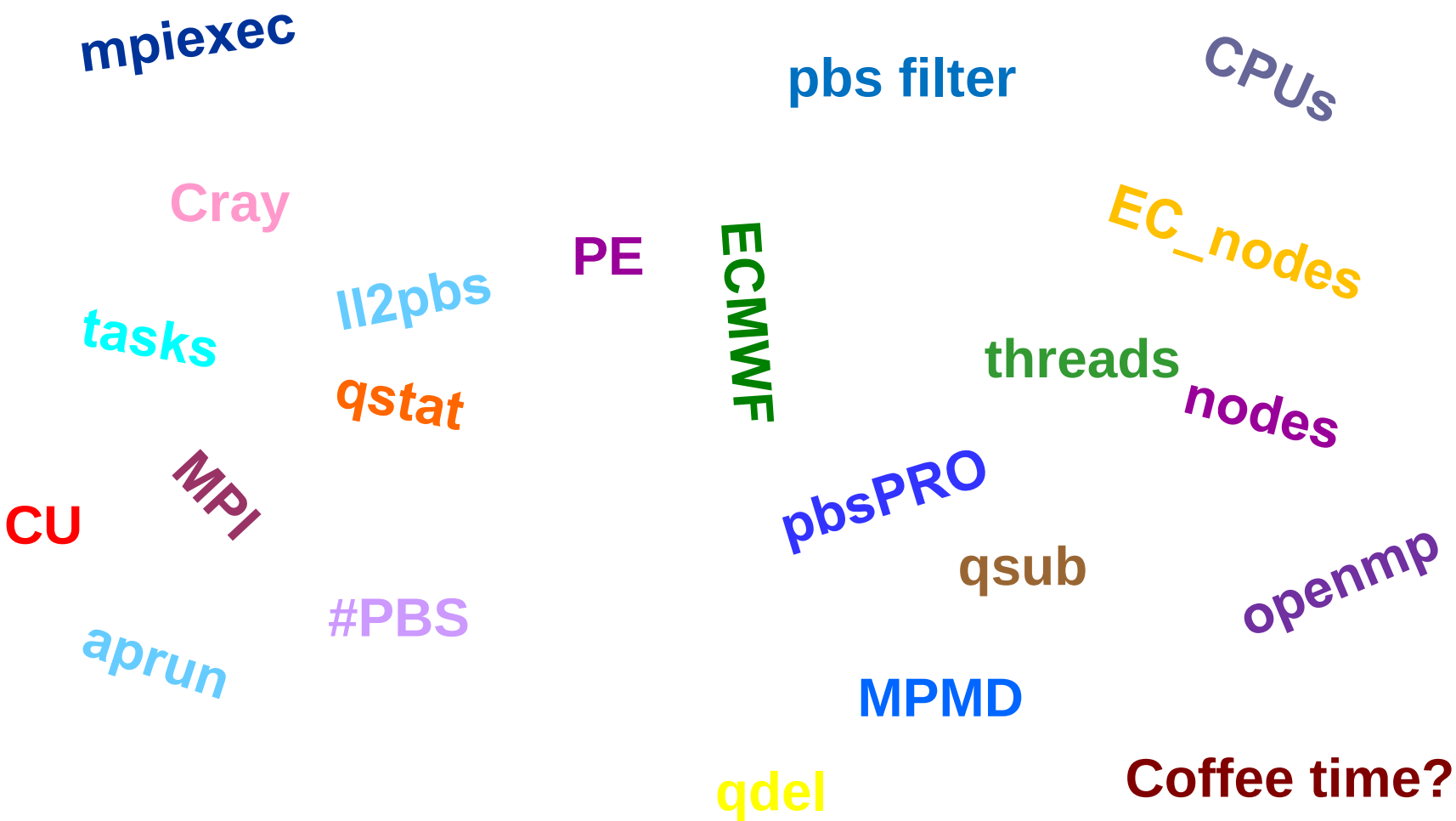
runs on 1 (full) node one XC30 but only on 2/3rd of a node on XC40. You should try to use a full node on XC40, i.e. request 72 tasks.

Cray upgrade to Phase II timeline (plan)

	Description
January 2016	OS upgrade ccb
February 2016	OS upgrade cca
February – March 2016	Extra LUSTRE file systems
March 2016	Install 1 extra cabinet on cca and ccb
March 2016	Replace 700 Ivy Bridge nodes with Broadwell nodes on ccb, followed by 7 weeks of testing
May 2016	Upgrade of remaining nodes on ccb and 3 weeks of testing
June 2016	Upgrade of cca to Broadwell nodes
July 2016	Phase II upgrade complete

- Subject to changes
- To be coordinated with the Operational upgrade to cycle 41r2 , due for 8 March 2016.

Question time …



Tutorial – on cca:

- cd
- tar xvf ~trx/pbs.tar
- cd pbs

'EC_' pbs directives (pages 15 and 17): Please submit the job 'pi-mpi-omp-nodes.cmd'. Can you adapt it to use the 'EC_' PBS directives defining the number of tasks and threads? Also try to use the corresponding 'EC_' variables in the job script. Try to submit the new job and check that it runs as the original job.

