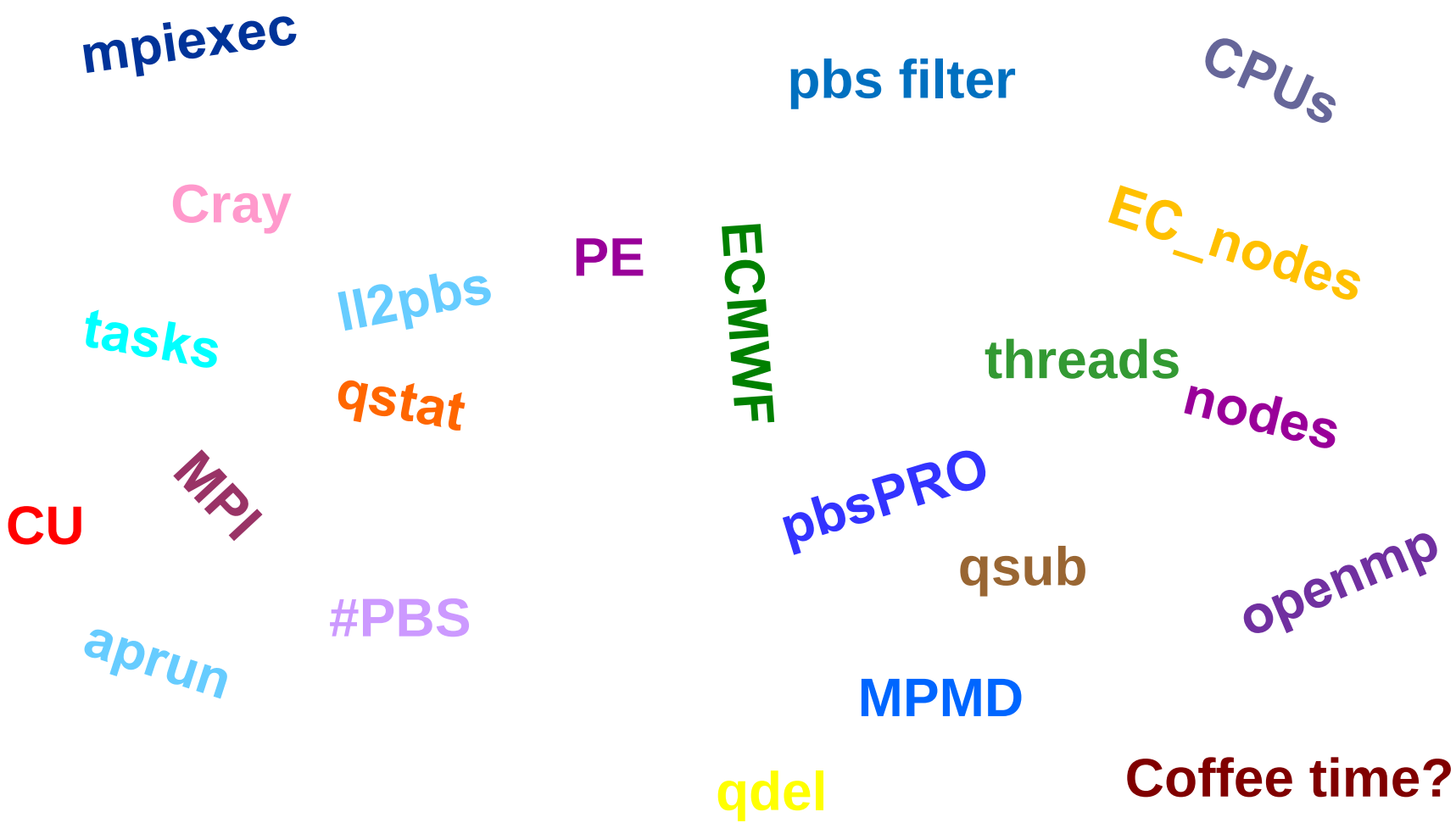
PBS(pro) at ECMWF

Dominique Lucas
User Support

Agenda



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

	Old	Current
Sustained performance	~70 teraflops	~ 210 teraflops
Peak performance	~1500 teraflops	~3480 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	64
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 aprun (equivalent of poe, mpirun, …) 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 <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.

- We used LoadLeveler on the previous IBM HPCF system.

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

LoadLeveler Job translation: ll2pbs

➢ ll2pbs –h

```
usage: ll2pbs [-h] [-q] [-s] [-i INSCRIPT] [-o OUTSCRIPT] [-f]

Job translator from LoadLeveler to PBSPro
```

➢ ll2pbs –i job_ll.cmd –o job_pbs.cmd

```
WARNING: directive cpu_limit not supported, skipping...
WARNING: No variables allowed in the output file definition in PBS.
Reverting to default values...
WARNING: No variables allowed in the error file definition in PBS.
Reverting to default values...
```

- Not all LoadLeveler directives exist in PBSpro
- ll2pbs is only an aiding tool, check resulting PBS job
- No change are made in script
- No change made in job geometry, no knowledge of queue ‘nf’
- PBSpro provides a ‘nqs2pbs’ command. See man page.

PBSpro environment variables

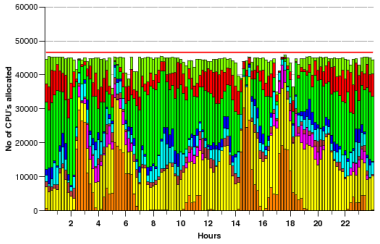
- A number of environment variables are set by PBSpro for all jobs:

```
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 …

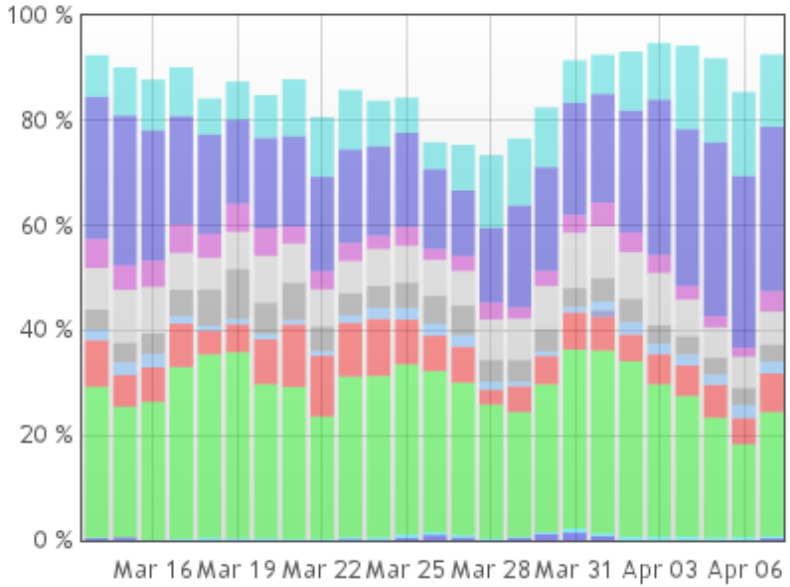
see later …

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

cca cluster node occupancy for 1 month: ~80%. In green, MS work. IBM (c2a) occupancy above: 94%



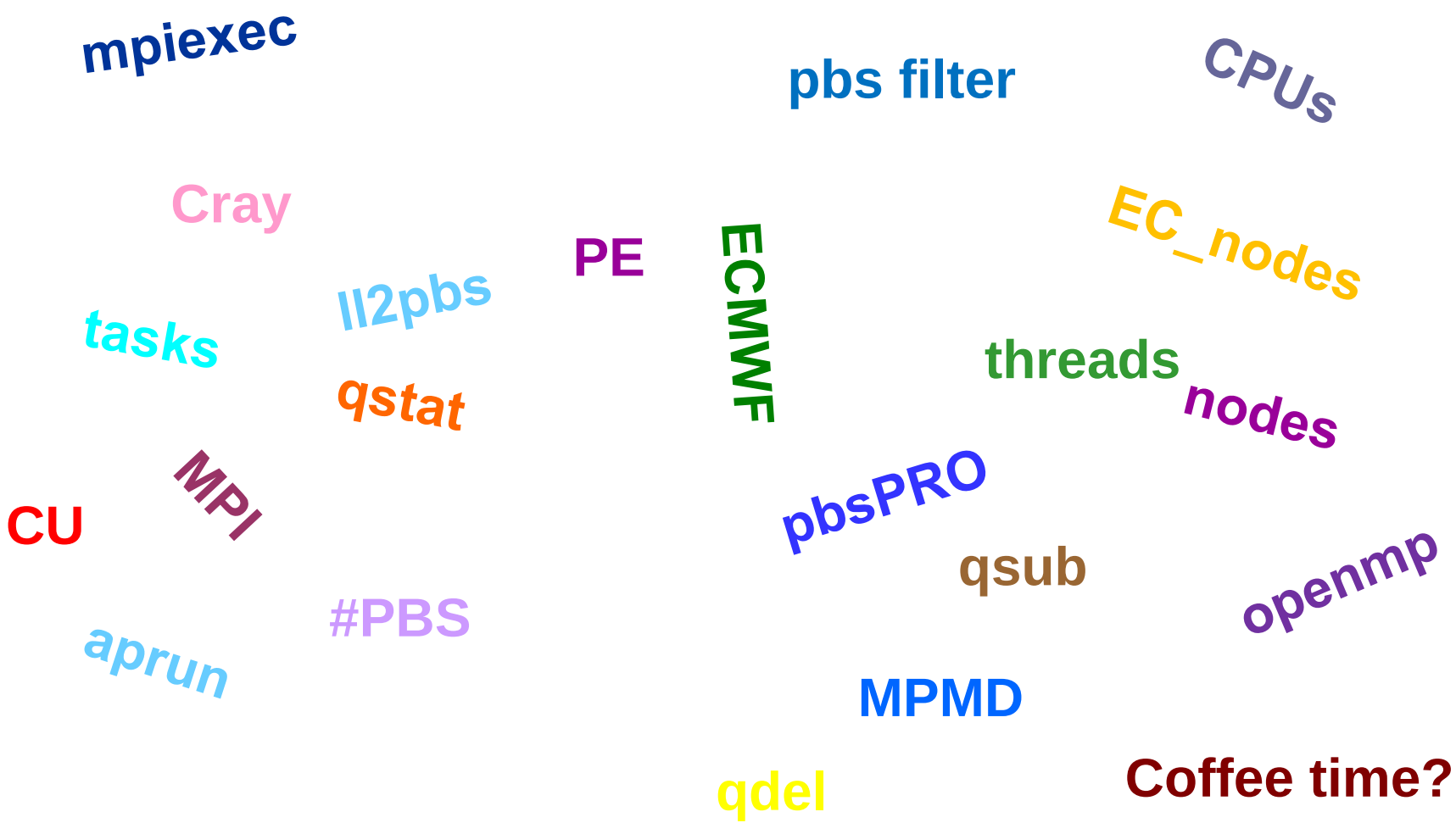
ECMWF's enhanced scheduling

- 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.
  - Use full path for output file name.

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 enhanced scheduling.
- Another complication 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 ‘select’ statement.
  - ECMWF PBS directives are available to define the job geometry.
  - Environmental variables defining the job geometry are available in the job.

Half time … questions?



Some more terminology (Cray vs. IBM)

- 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 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

➢ 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
…
```


ECMWF job example 4 – fractional job

➢ cat small_job.cmd

```
..
#PBS -N HelloMPI_S
#PBS -q nf
#PBS -l EC_total_tasks=12
#PBS -l EC_hyperthreads=2
…
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 covers all the user requirements.
  - The Hook 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

- LoadLeveler's basic work unit is a "job step“
  - serial/parallel steps can be mixed in a job.
- 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.

Multi step jobs - PBSpro

- ‘qsub –W depend’ option

➢ cat prepare_run.cmd

…

echo “Start by submitting model run and post-processing jobs’

MPI_JOBID=`qsub -W depend=afterok:$PBS_JOBID run_model.cmd`

qsub -W depend=afterok:$MPI_JOBID post_proc.cmd

echo “Now the preparation of the run follows, e.g. compilations, ECFS, MARS”

…

➢ qsub prepare_run.cmd

124938.sdb

➢ qstat –u xyz

Job ID	Username	Queue	Jobname	Req'd NDS	Req'd TSK	Status
124938.sdb	xyz	ns	prepare_run	1	1	R
124939.sdb	xyz	np	run_model	3	49	H
124940.sdb	xyz	ns	post_proc	1	1	H

Multi step jobs - PBSpro

- ‘qsub –W block=true’ option

➢ cat pre_and_post.cmd

```
…
echo ‘Preparation work …”
…
qsub -W block=true run_model.cmd
echo “Now the post-processing”
…
```

➢ qsub pre_and_post.cmd

124944.sdb

➢ qstat –u xyz

Job ID	Username	Queue	Jobname	Req'd NDS	Req'd TSK	Status
124944.sdb	xyz	ns	pre_and_post	1	1	R
124945.sdb	xyz	np	run_model	3	49	R

- See ‘man qsub’ for more information

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 …
```

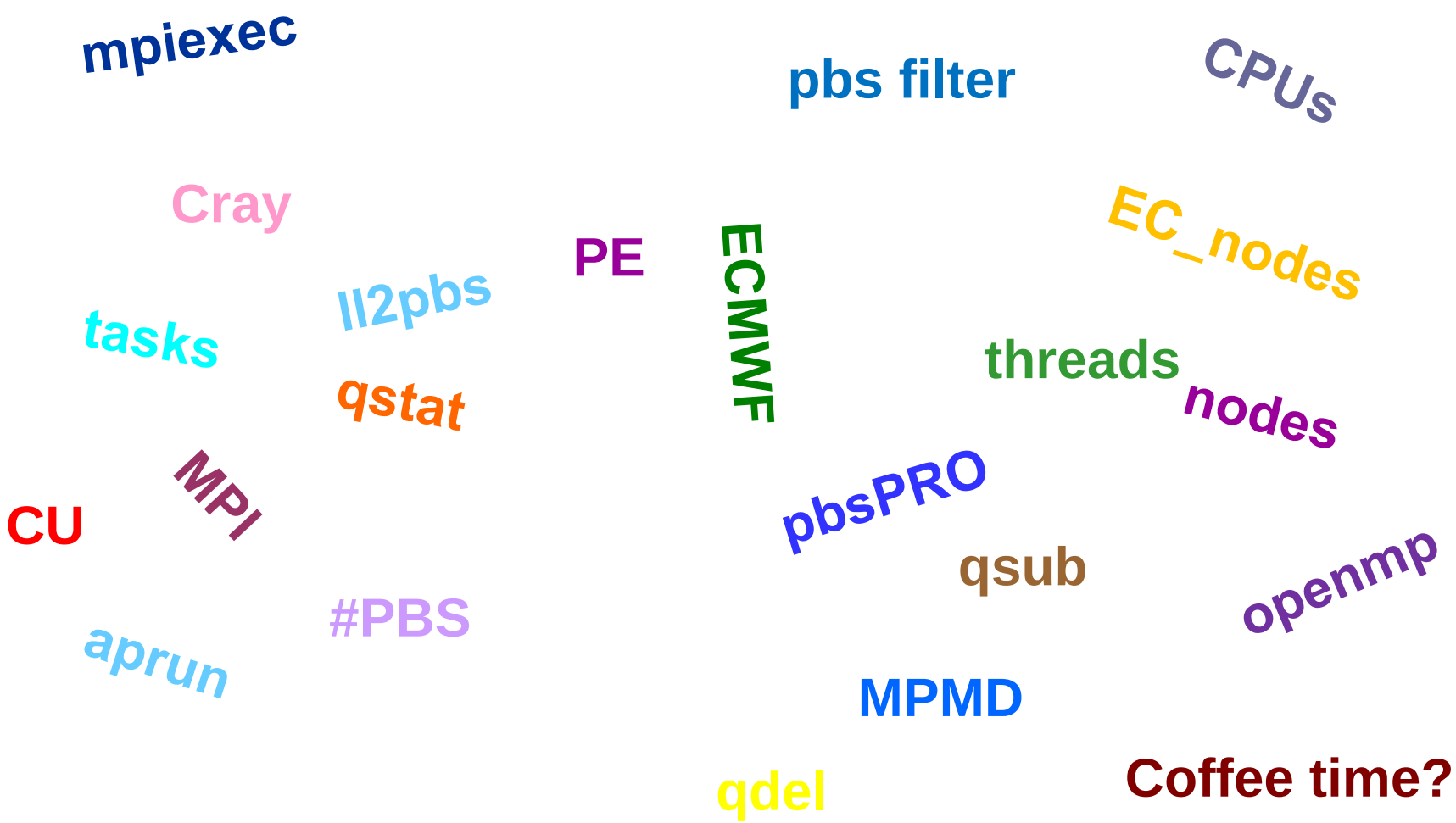
- From remote systems, via ECaccess, to start GUI applications:

```
➢ 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
```


Future changes

- $TMPDIR:
  - Currently on LUSTRE, under $TEMP.
  - We plan to make it available in memory, due to inefficient I/O, e.g. for compilations.
- Multi complex PBS:
  - Transparent job submission to cca/ccb
  - Sharing of job scheduling between different PBS ‘servers’.
  - Better utilisation of the two machines
- Enforcement of job memory limits.

Question time …



Tutorial – on cca

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

1. 'EC_' pbs directives (page 17): Please submit the job 'pi-mpi-omp-nodes.cmd'. Can you adapt it to use the EC_ directives defining the number of tasks and threads. Also use the EC_ variables in the job. Submit the new job and make sure it runs.

2. Job steps (page 22): Please adapt the job 'compile_code.cmd' to submit the jobs 'pi-mpi.cmd' and 'put_ecfs.cmd' using the correct dependencies.

3. (Optional) ll2pbs (page 9): Please translate the LoadLeveler job 'pi_ll.cmd' to a pbs job. Check the pbs job (including compilation statement!) and submit the job.

