Using Qbox on ANL Theta

François Gygi
University of California, Davis
fgygi@ucdavis.edu
http://eslab.ucdavis.edu
http://qboxcode.org
http://www.quantum-simulation.org

Apr 21 2021

UCDAVIS UNIVERSITY OF CALIFORNIA

MICCoM

Qbox documentation

Online documentation:

http://qboxcode.org

http://qboxcode.org/doc/html

Compiled versions on theta: in the group project directory:

/lus/eagle/projects/LightActivMat/qbox

SG15 potentials on theta:

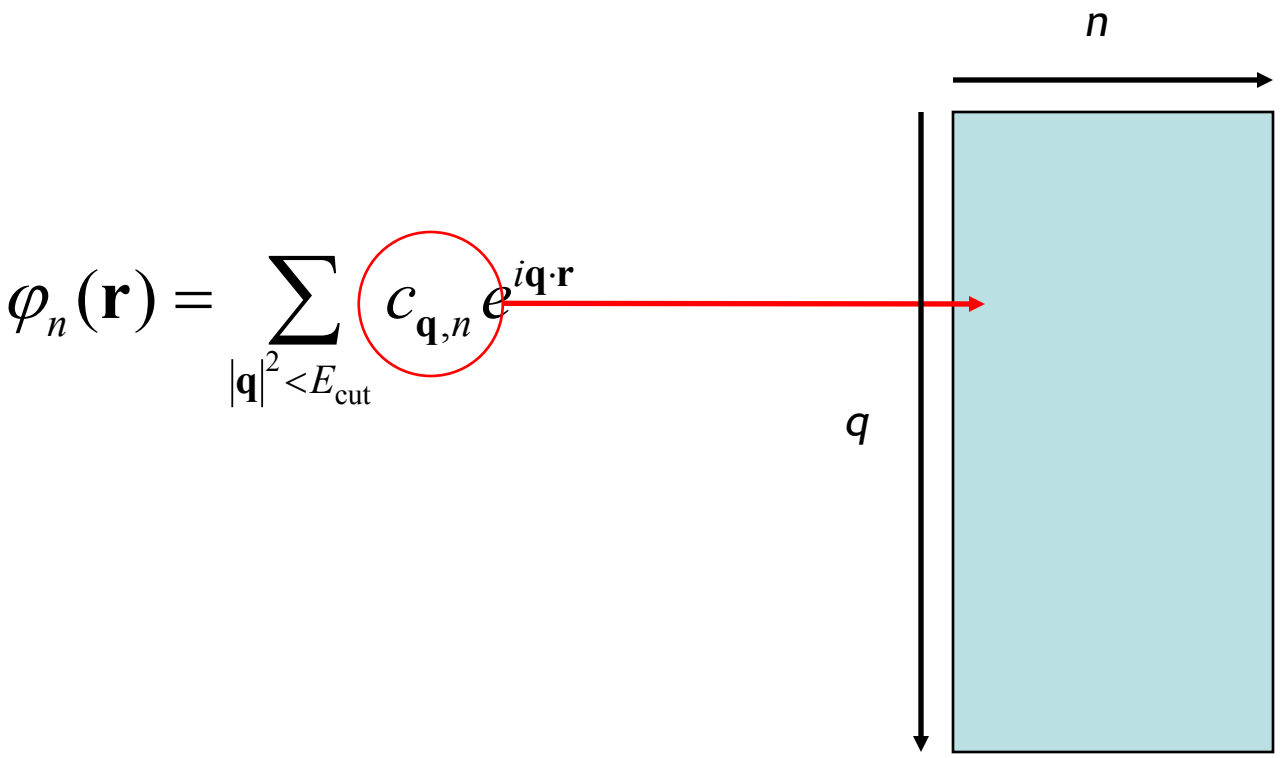
/lus/eagle/projects/LightActivMat/qbox/potentials/sg15/xml

UCDAVIS UNIVERSITY OF CALIFORNIA

MICCoM

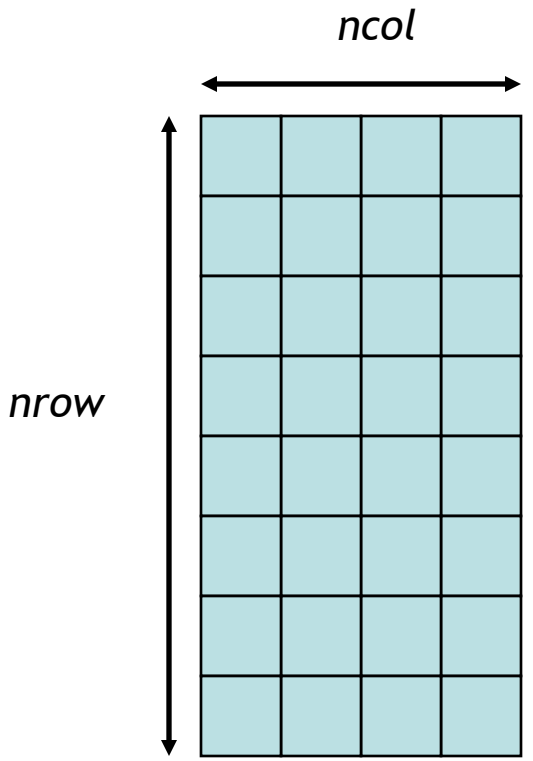
Qbox data layout

- Matrix of plane wave coefficients c_{qn}



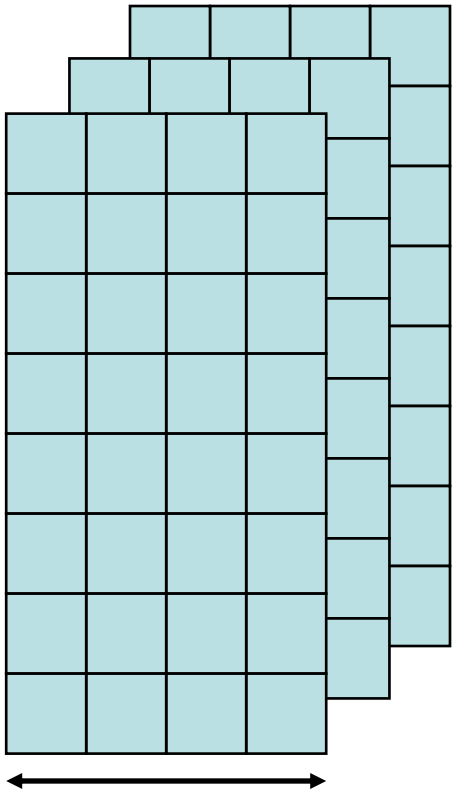
Qbox task allocation

- The matrix of coefficients is block-distributed among MPI tasks
- Until version 1.71.x: use the *nrowmax* variable to control task allocation
- *nrow* * ncol = nprocs
- *nrow* <= *nrowmax*
- all k-points and spins on the same task



Qbox task allocation (1.72.x +)

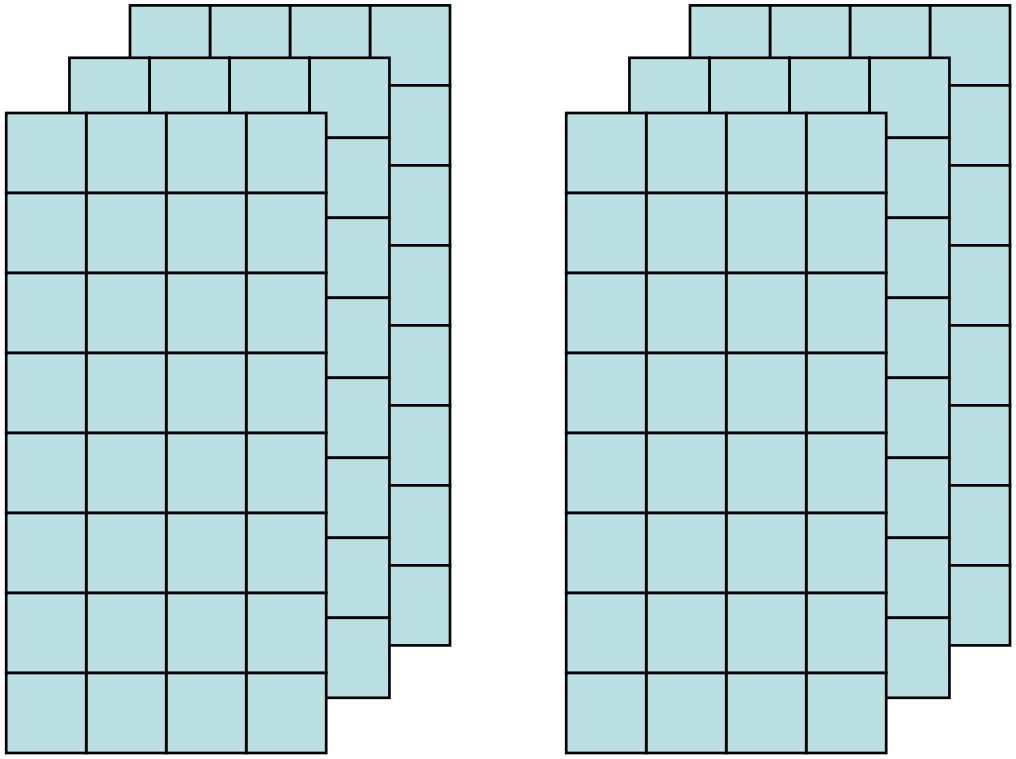
- Since version 1.72.x: use command-line arguments to control task allocation
- *-nstb : # of state blocks*
- *-nkpb : # of k-point blocks*
- *-nspb : # of spin blocks*
- k-points and spins can be distributed on separate tasks
- example: nproc=96



Qbox task allocation (1.72.x +)

- nprocs=96, nstb=4, nkpb=3, nspb=2

nkpb=3



up spin *dn spin*

Task allocation on theta

- Theta nodes have 64 cores
- Possible uses:
  - 64 MPI tasks/node, 1 thread/task
  - 64 MPI tasks/node, 2 threads/task
  - 32 MPI tasks/node, 2 threads/task
  - 32 MPI tasks/node, 4 threads/task
- Optimal choice requires trial and error
- Splitting wave functions across nodes degrades performance significantly

Theta queues

- The Theta debug queue is limited to 8 nodes, 1hr (not for production!)
- Next size is 128 nodes (!)
  - long wait time (> 1 day)
  - only 3 hours job duration
- Larger queues
  - >=256 nodes, 6 hr
  - >=384 nodes, 9 hr
  - >=640 nodes, 12 hr
  - >=802 nodes, 24 hr

Qbox jobs

- Example **`qbox_theta.job`**
- Copy the job script **`qbox_theta.job`** to **`myrun.job`**
- Edit the job script to adjust parameter, # of nodes, etc.
- Prepare Qbox input in **`myrun.i`**
- **`$ qsub myrun.job`**
- Qbox results will be written on **`myrun.r`**

```
#!/bin/bash
#COBALT -t 0:30:00
#COBALT -n 8
#COBALT -q debug-cache-quad
#COBALT -A LightActivMat
exe=/lus/eagle/projects/LightActivMat/qbox/rel1_73_1/src/qb
scriptname=$0
file=${scriptname%.*}
export n_nodes=$COBALT_JOBSIZE
export n_mpi_ranks_per_node=64
export n_mpi_ranks=$(($n_nodes * $n_mpi_ranks_per_node))
export OMP_NUM_THREADS=1
#export OMP_PLACES=cores
#export OMP_PROC_BIND=true
export QBOX_OPTS='-nstb 4'
aprun -n $n_mpi_ranks -N $n_mpi_ranks_per_node -d 1 -j 1 -
cc depth \
$exe $QBOX_OPTS $file.i > $file.r
```

Ensemble Qbox jobs

- Most Qbox jobs will not scale well on 128 theta nodes (8192 cores)
- If multiple similar runs can be used to gather statistics, use ensemble jobs
- If using SSAGES, create multiple walkers on separate nodes
- Use one **`aprun`** command for each Qbox instance
- End the **`aprun`** command line with **`'&'`**
- Add **`wait`** command at end

```
#!/bin/bash
#COBALT -t 0:30:00
#COBALT -n 2
#COBALT -q debug-cache-quad
#COBALT -A LightActivMat
exe=/lus/eagle/projects/LightActivMat/qbox/rel1_72_3/src/qb
export OMP_NUM_THREADS=2
aprun -n 32 -N 32 -d 2 -j 1 -cc depth \
  $exe -nstb 1 qbox_0.i > qbox_0.r &
aprun -n 32 -N 32 -d 2 -j 1 -cc depth \
  $exe -nstb 1 qbox_1.i > qbox_1.r &
wait
```