STEADY STATE GENETIC ALGORITHM FOR CRYSTAL STRUCTURE PREDICTION: A SCALABLE APPROACH FOR THE OSG

Varela K. N.[1], Pagola G. I.[1], Ferraro M. B.[1], Lund A. M.[2], Orendt A. M.[3], Facelli J. C.[2]

[1]Physics Department, FCEyN (UBA) - IFIBA – CONICET.

[2]Department of Biomedical Informatics, University of Utah, Salt Lake City, US.

[3]Center for High Performance Computing, University of Utah, Salt Lake City, US.

UBA Universidad de Buenos Aires
Argentina virtus robur et studium

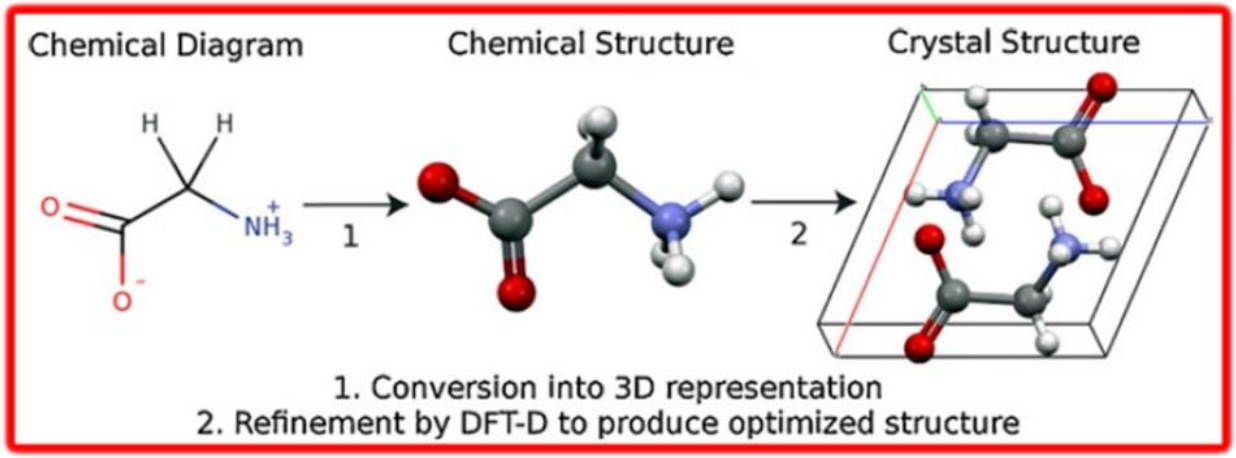
Universidad de Buenos Aires - Exactas
departamento de física

The University of Utah
Biomedical Informatics

Overview

- Crystal Structure Prediction (CSP)
- Steady State Genetic Algorithm for CSP
- Why OSG for our project
- Job distribution on OSG
- Research progress and next steps

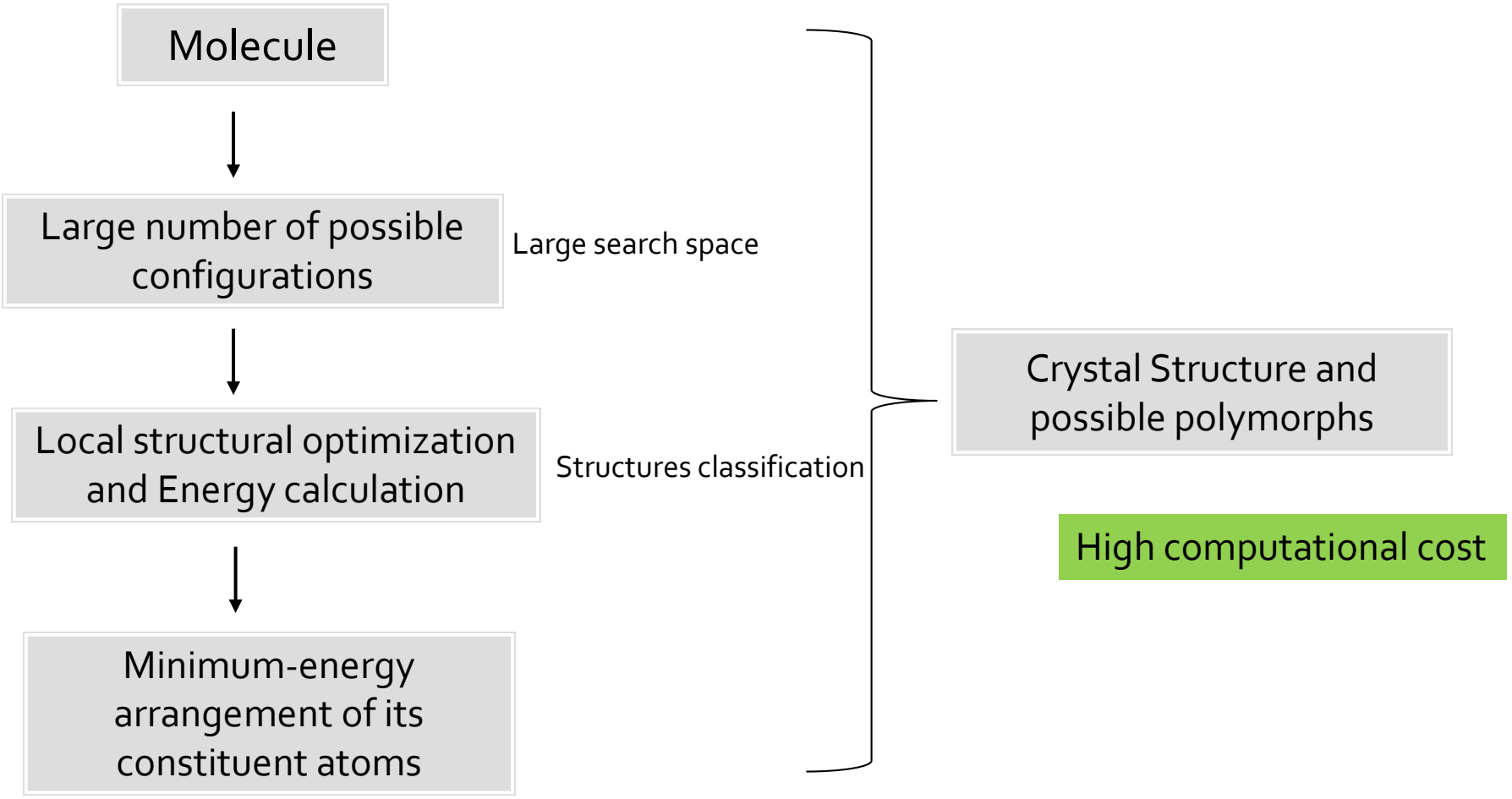
Crystal Structure Prediction (CSP)



- New compounds and materials
- Crystal and co-crystal design
- Crystallization process
- Polymorphism

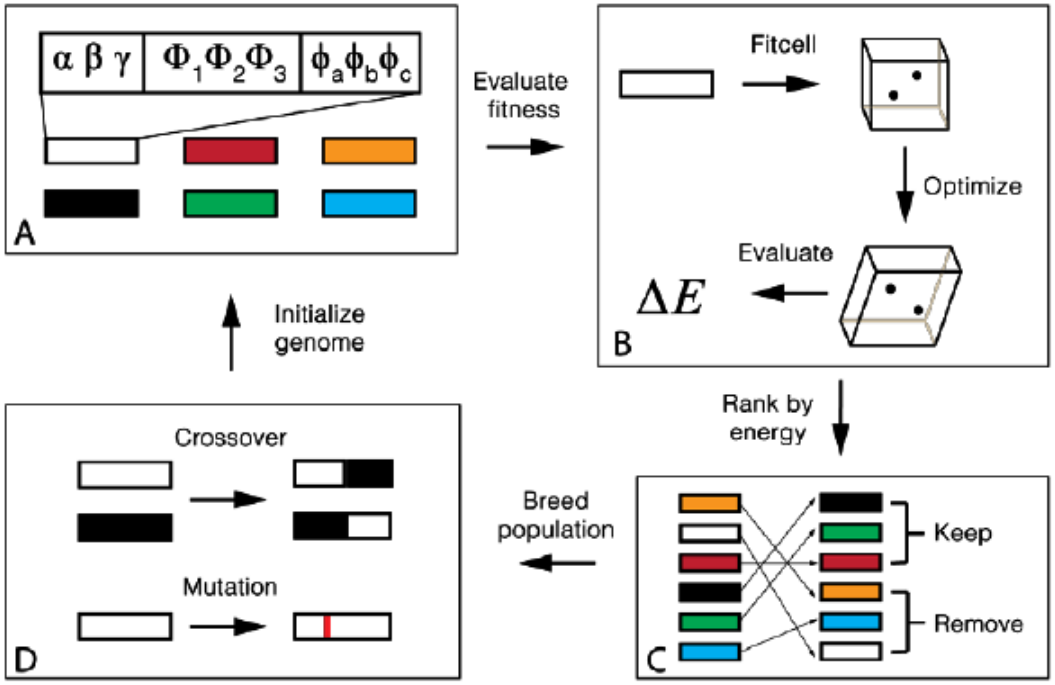
Albert M. Lund, Gabriel I. Pagola, Anita M. Orendt, Marta B. Ferraro, Julio C. Facelli, Crystal structure prediction from first principles: The crystal structures of glycine, Chemical Physics Letters, Volume 626, 2015, Pages 20-24,

Crystal Structure Prediction (CSP)



Steady State Genetic Algorithm for CSP

MGAC (*Modified Genetic Algorithm for Crystals) Processing Scheme*



- Random initial population
- Energy of the candidate structure is the selection criteria for best structure
- Specific physics parameters are optimized

Quantum-espresso (QE-DFT) Software is used for energy calculation and optimization

Lund 2010, Crystal Growth & design
Kim 2008, j Comp Chem

Why OSG for our project?

Advantage of using OSG

- **Distribution of QE-DFT calculations**
- High performance nodes
- Efficiency
- **Distribution of calculations independently**
- OSG is the only open science architecture that scales to the level needed for first principle CSP

Jobs distribution on the OSG

MGAC3

1. Random population (Random "genes")
2. List of Best Structures (lowest energy)
3. Incorporation to the list of best structures according to the ground state energy, or discarded.
4. *Periodically, recombination and generation of new children*
5. 3. and 4. are repeated periodically

Initial population - QE's inputs

QE's outputs

Children - QE's inputs

Open Science Grid

- Jobs Distribution
- QE-DFT calculations

Jobs distribution on the OSG

Open Science Grid

- Distribution
- QE-DFT calculations

First jobs on OSPool

Executable

```
#!/bin/bash
source /cvmfs/oasis.opensciencegrid.org/osg/modules/lmod/5.6.2/init/bash
module load espresso
date
ls -l
printf "Start time: "; /bin/date
printf "Job is running on node: "; /bi
printf "Job running as user: "; /usr/b
printf "Job is running in directory: "

pw.x -inp  INPUT_FILE.in > OUTPUT_FILE.out
```

QE espresso was in a pre-installed software packages

Submit file

```
universe = vanilla
+ProjectName = "osg.SSGAforCSP"
Error = job.$(Cluster).$(Process).error
Output = job.$(Cluster).$(Process).out
log = job.$(Cluster).$(Process).log

# OS version, core count and memory, and wants to use the software modules.
Requirements = OSGVO_OS_STRING == "RHEL 6" && Arch == "X86_64" && HAS_MODULES == True
request_cpus = 1
request_memory = 1 GB


transfer_input_files =  sys_qe_spcgrp4n128.in,C.pbe-rrkjus.UPF,O.pbe-rrkjus.UPF,H.pbe-rrkjus.UPF

# requirements = (HAS_CVMFS_oasis_opensciencegrid_org =?= TRUE)

executable = em-QE28657.sh
Arguments = $(Cluster)

queue 1
```

Jobs were sent one by one

Jobs distribution on the OSG

Jobs submission optimization

- **Quantum-espresso is now running as a container**

```
universe = vanilla
+ProjectName = "osg.SSGAforCSP"
Error = job.$(Cluster).$(Process).error
Output = job.$(Cluster).$(Process).out
log = job.$(Cluster).$(Process).log

# OS version, core count and memory, and wants to use the software modules.

Requirements = (HAS_SINGULARITY == TRUE) && (IsOsgVoContainer =!= true)

+SingularityImage = "/cvmfs/singularity.opensciencegrid.org/opensciencegrid/osgvo-quantum-espresso:6.8"
```

- **Jobs are now submitted through a list**

```
transfer_input_files = $(inputfile),C.pbe-rrkjus.UPF,O.pbe-rrkjus.UPF,H.pbe-rrkjus.UPF
transfer_output_files = $(outputfile)
transfer_checkpoint_files = check.checkpoint
checkpoint_exit_code = 85

executable = executable.sh
arguments = $(inputfile) $(outputfile)

queue inputfile, outputfile from list.txt
```

```
sys_qe_spcgrp1n68.in, pru.sys_qe_spcgrp1n68.out
sys_qe_spcgrp4n69.in, pru.sys_qe_spcgrp4n69.out
sys_qe_spcgrp7n70.in, pru.sys_qe_spcgrp7n70.out
sys_qe_spcgrp1n71.in, pru.sys_qe_spcgrp1n71.out
sys_qe_spcgrp1n72.in, pru.sys_qe_spcgrp1n72.out
sys_qe_spcgrp4n73.in, pru.sys_qe_spcgrp4n73.out
sys_qe_spcgrp1n74.in, pru.sys_qe_spcgrp1n74.out
```

Jobs distribution on the OSG

Jobs submission optimization

- **Memory requirement optimization**

```
request_cpus = 1
request_disk = 1 GB
request_memory = ifthenelse(isundefined(NumHolds) || NumHolds == 0, 6000, 10000)
# if the jobs is held due to memory issue, allow it to restart with increased
# memory request (see above)
periodic_hold = MemoryUsage > RequestMemory
periodic_release = (JobStatus == 5) && (NumHolds < 2) && ( (HoldReasonCode == 26) || (HoldReasonCode == 3) )
```

- **MPI Issues - Self Checkpoint implementation**

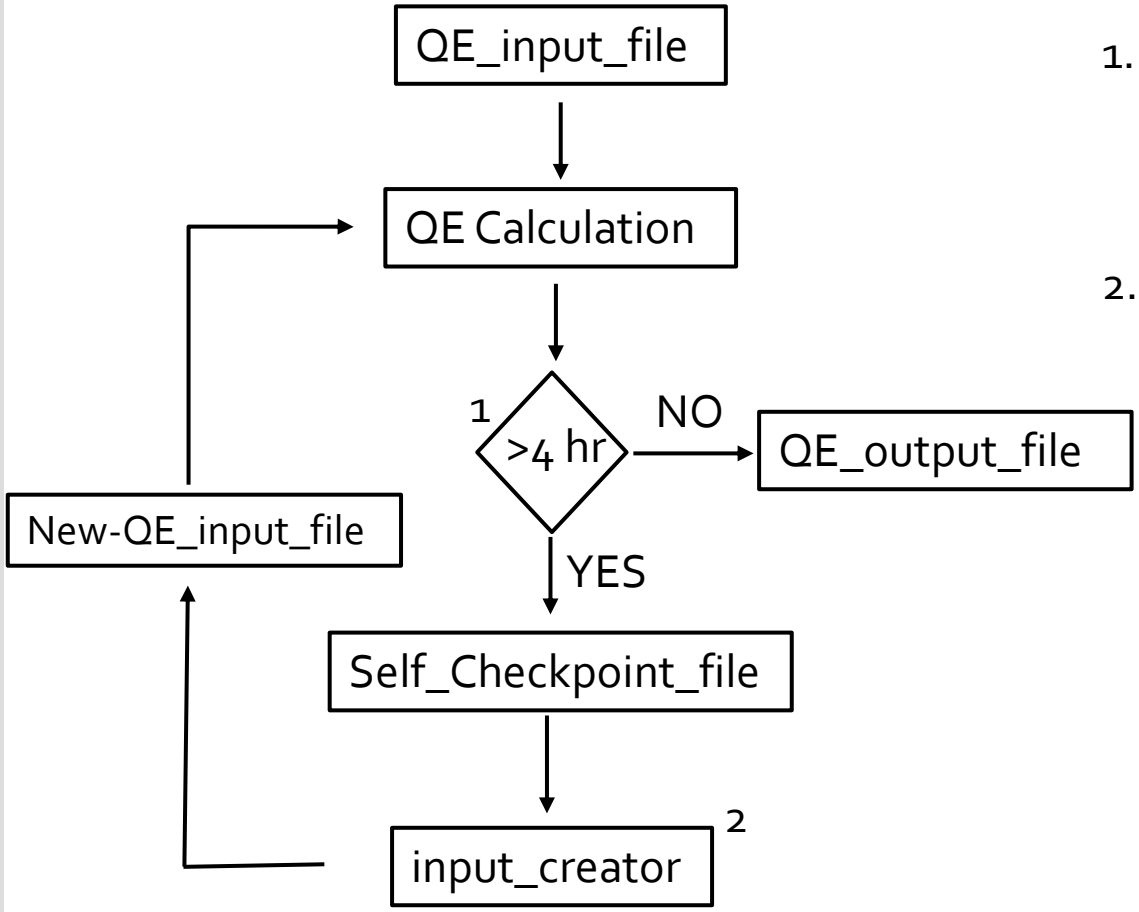
```
transfer_input_files = $(inputfile),C.pbe-rrkjus.UPF,O.pbe-rrkjus.UPF,H.pbe-rrkjus.UPF, input_creator
transfer_output_files = $(outputfile)
transfer_checkpoint_files = check.checkpoint
checkpoint_exit_code = 85

executable = executable.sh
arguments = $(inputfile) $(outputfile)
```

For long jobs or if the job is taken out from de running node, it is now availed to start from this point even in another node.

Jobs distribution on the OSG

Jobs submission optimization

- **Self Checkpoint implementation.**



1. We set a safe stop (on QE_input_file) every 4 hours to make a checkpoint

2. When a checkpoint is generated, a new input need to be created, we use an extra algorithm in C++ to accomplish this (we took the last atomic positions calculated by QE to start from there)

Jobs distribution on the OSG

Jobs submission optimization

- **Self Checkpoint implementation**

```
#!/bin/bash
FILEIN="$1"
FILEOUT="$2"
FILECHECK="check.checkpoint"


./input_creator $FILECHECK
exit_ejec=$?

if [ "$exit_ejec" == 99 ]
then
echo ".sh: entro a 99"
pw.x -inp  $FILEIN >tmpqe.out
exit_QE=$?
elif [ "$exit_ejec" == 101 ]
then
echo ".sh: Not enough space allocated for radial FFT(TO CHECKPOINT)"
pw.x -inp  new_input.in >tmpqe.out
exit_QE=$?
fi

echo "print_exit_code QE"
echo $exit_QE
```

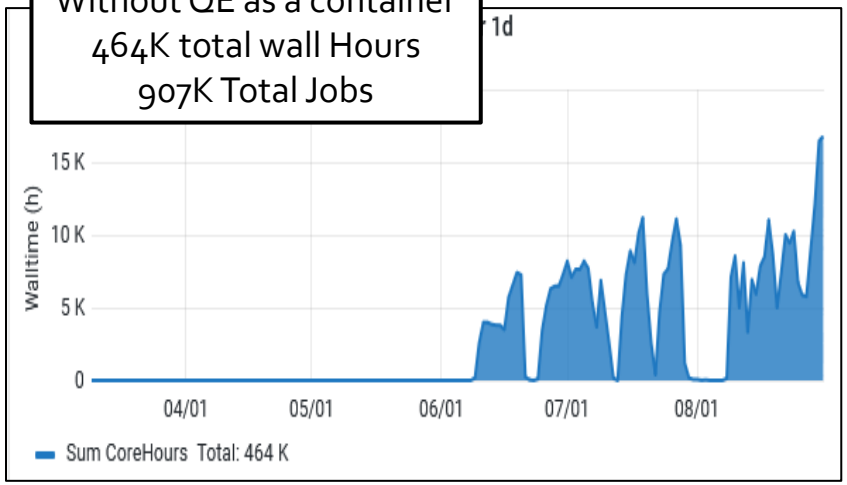
First calculation.

Some QE-issue that need to be addressed with checkpoint approach.

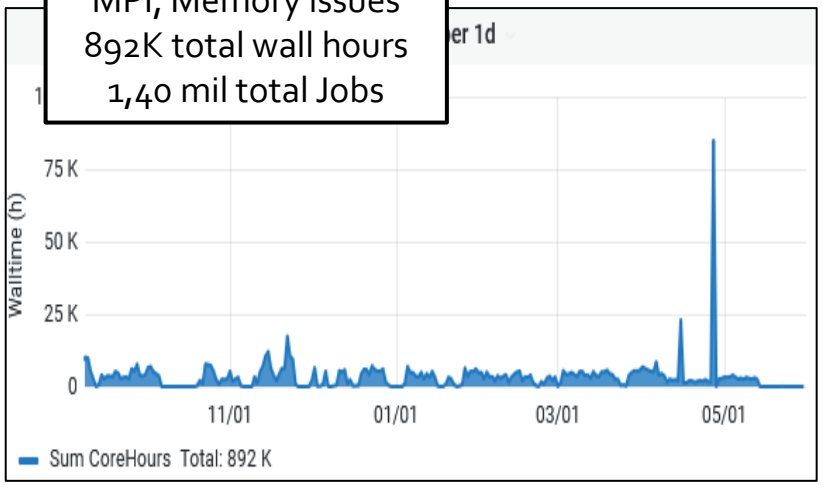
```
if [ "$exit_QE" == "1" ]
then
echo ".sh: QE stop 1. QE-ERROR-EXIT"
exit 0
fi
if [ "$exit_QE" == "2" ]
then
echo ".sh: max_seconds - to checkpoint"
exit 85
fi
if [ "$exit_QE" == "3" ]
then
echo ".sh: nstep - to checkpoint"
exit 85
fi
if [ "$exit_QE" == "0" ]
then
mv $FILECHECK $FILEOUT
echo "JOB DONE"
exit 0
```

Jobs distribution on the OSG

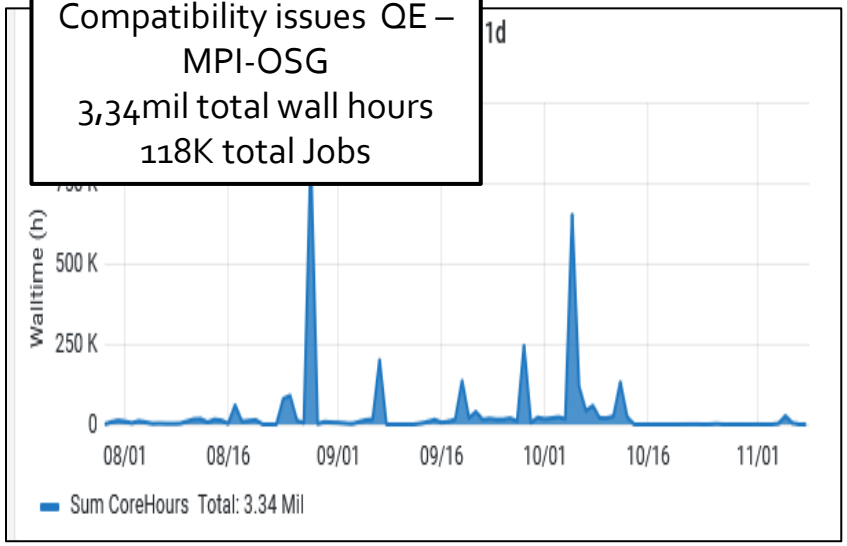
Without QE as a container
464K total wall Hours
907K Total Jobs



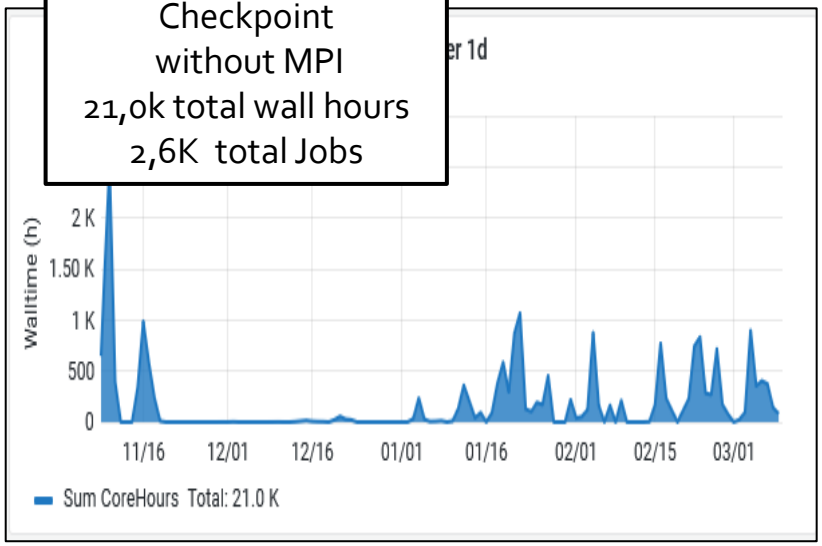
MPI, Memory issues
892K total wall hours
1,40 mil total Jobs



Compatibility issues QE – MPI-OSG
3,34mil total wall hours
118K total Jobs



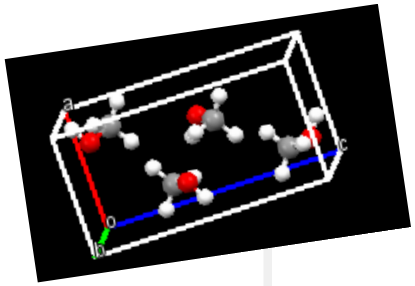
Checkpoint
without MPI
21,0k total wall hours
2,6K total Jobs



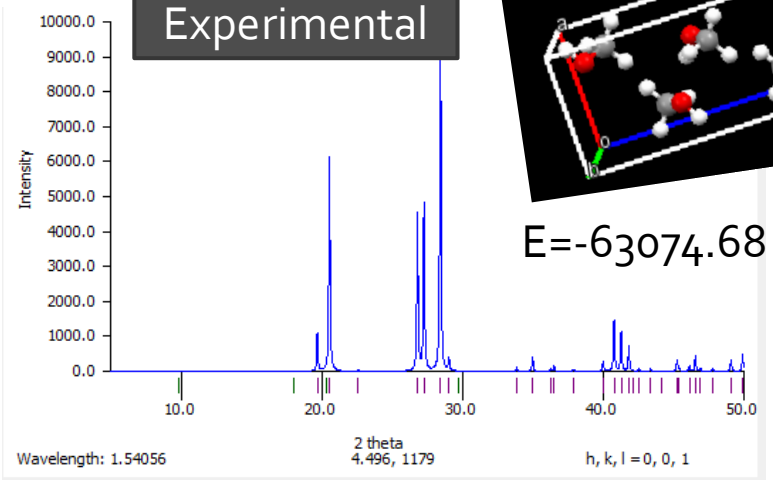
Research progress and next steps

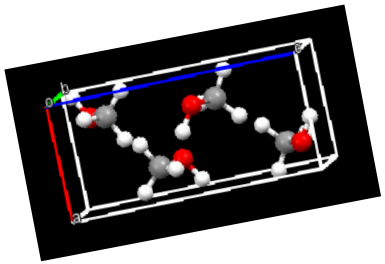
Test structure: Methanol

Experimental

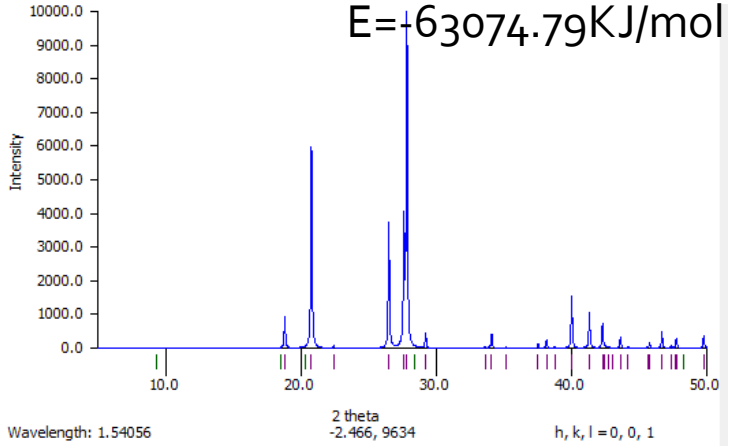


E=-63074.68 KJ/mol

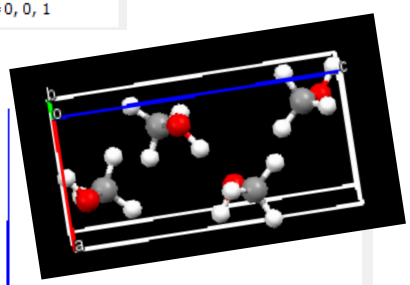




E=-63074.79KJ/mol



MGAC



E=-63074.61 KJ/mol

MGAC

Research progress and next steps

- For the first structural approximation (Methanol molecule):
  Evaluated structures: ≈ 20,000
  QE calculation average per structure: ≈ 12 core hour
  Total time spent: ≈ 260k core hour

- The MGAC effectively achieving convergence to the desired objective (a particular structure or set of features comes to dominate the population).

- Implementations to improve the use of OSG resources (e.g., self checkpoint) takes time, but also significantly improves the performance of the running jobs

- We are making some improvements to our code and at the same time optimizing the execution of the jobs with the checkpoint implementation.

- Change the test molecule to another one and explore the possible configurations to get the crystalline structure and possible polymorphs

OSG support

- OSG Virtual School 2021
  Tools and resources available

- OSG staff
  Optimization of the OSG resources. What is the best tool for each research

Thanks