USING VASP ON ANVIL

Nannan Shan, PhD

Sr. Computational Scientist

09/15/2023

PURDUE UNIVERSITY | Rosen Center for Advanced Computing

Code of conduct

This external code of conduct for ACCESS-sponsored events represents ACCESS's commitment to providing an inclusive and harassment-free environment in all interactions regardless of race, age, ethnicity, national origin, language, gender, gender identity, sexual orientation, disability, physical appearance, political views, military service, health status, or religion. The code of conduct below extends to all ACCESS-sponsored events, services, and interactions.

Webpage: https://support.access-ci.org/code-conduct

How to Submit a Report

If you feel your safety is in jeopardy or the situation is an emergency, contact local law enforcement before making a report to ACCESS. (In the U.S., dial 911.)

ACCESS is committed to promptly addressing any reported issues. If you have experienced or witnessed behavior that violates the ACCESS Code of Conduct, please submit a ticket to ACCESS by using this online form.

Acknowledgements

- "This material is based upon work supported by the National Science Foundation under Grant No. 2005632."

- ***Disclaimer***: "Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation."

What is VASP?

- The Vienna *Ab initio* Simulation Package (VASP) is a computer program for atomic scale materials modelling, e.g. electronic structure calculations and quantum-mechanical molecular dynamics, from first principles.
- VASP computes an approximate solution to the **many-body Schrödinger equation**, either within *density functional theory* (DFT), solving the Kohn-Sham equations, or within the *Hartree-Fock* (HF) approximation, solving the Roothaan equations. *Hybrid functionals* that mix the Hartree-Fock approach with density functional theory are implemented as well. Furthermore, *Green's functions methods* (GW quasiparticles, and ACFDT-RPA) and *many-body perturbation theory* (2nd-order Møller-Plesset) are available in VASP.[1]

[1] www.vasp.at

Agenda

- VASP access on Anvil
- Available VASP builds on Anvil
- Compiling VASP on Anvil
- Running VASP on Anvil
- Known errors users run into
- Best practice

VASP on Anvil

VASP ACCESS

VASP Access

- VASP access is controlled by unix groups on Anvil.
- Bring your own VASP license to use VASP on Anvil.
- To request VASP access, please send a ticket to ACCESS Help Desk to request access and provide your ***registered email*** associated with VASP license.
- Prospective users can use the command below to check their unix groups on Anvil.

```
@login02.anvil:[~] $ groups
x-acs00000 vasp6 vasp5
```

VASP on Anvil

Available VASP builds

Available VASP modules on Anvil

Currently, Anvil provides **VASP/5.4.4.pl2** and **VASP/6.3.0** modules with our default environment compiler gcc/11.2.0 and mpi library openmpi/4.0.6.

You can use the **VASP/5.4.4.pl2** module by:

```
$ module load gcc/11.2.0  openmpi/4.0.6
$ module load vasp/5.4.4.pl2
```

You can use the **VASP/6.3.0** module by:

```
$ module load gcc/11.2.0  openmpi/4.0.6
$ module load vasp/6.3.0
```

VASP5 / VASP6 modules on Anvil

- Once a VASP module is loaded, you can choose one of the VASP executables to run your code:

```
vasp_std
vasp_gam
vasp_ncl
```

- The VASP pseudopotential files are not provided on Anvil, you may need to bring your own **POTCAR** files.

Note: only license-approved users can load the VASP module file.

VASP on Anvil

Build your own VASP

Build your own VASP on Anvil

- If you would like to use your own VASP on Anvil, please follow the instructions for Installing VASP.6.X.X and Installing VASP.5.X.X from VASP website:

  www.vasp.at/wiki/index.php/Installing_VASP.6.X.X
  www.vasp.at/wiki/index.php/Installing_VASP.5.X.X

- We also provide instructions about how to install VASP5 and VASP6 with GCC+openmpi or Intel+impi on Anvil and installation scripts:

  VASP 5: www.rcac.purdue.edu/knowledge/anvil/software/installing_applications/vasp/build_your_own_vasp_5
  VASP 6: www.rcac.purdue.edu/knowledge/anvil/software/installing_applications/vasp/build_your_own_vasp_6

VASP on Anvil

Running VASP

VASP input files

VASP requires 4 input files to run a calculation:

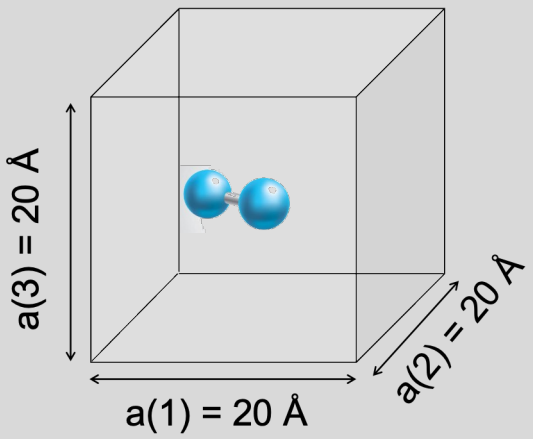
- POSCAR
- INCAR
- KPOINTS
- POTCAR

Example: O_2 molecule in a box

VASP Inputs files – POSCAR

POSCAR is a plain text file and contains at least the lattice geometry and the ionic positions, optionally, also starting velocities for a molecular-dynamics simulation.

```
O2 molecule in a box        ! header (comment)
1.0                         ! scaling parameter
20.0  0.0  0.0              ! Bravais matrix
0.0  20.0  0.0
0.0  0.0  20.0
O                           ! Name(s) of atomic type(s)
2                           ! Number of atoms (of each type)
Selective Dynamics          ! (optional: selective dynamics)
Cartesian                   ! Cartesian or Direct coordinates
10.7 10 11.52  T  T  T      ! positions of each atom
10.7 10 10.28  T  T  T
```



https://www.vasp.at/wiki/index.php/POSCAR

VASP Inputs files – INCAR

INCAR determines what to do and how to do it.

- It contains tags that select the algorithms and set the parameters.
- The settings in the INCAR file are the main source of errors and false results, so carefully checking the meaning of the set INCAR tags.

```
SYSTEM = O2 in a box
GGA = PE
ISMEAR = 0
ISPIN = 2
```

For a list of all INCAR-tags have a look at:
https://www.vasp.at/wiki/index.php/Category:INCAR_tag

VASP Inputs files – KPOINTS

KPOINTS specify the points VASP will use to sample the first Brillouin zone in reciprocal space.

```
Automatic mesh      ! Header (comment)
0                   ! determine number of k points automatically
G or M              ! Γ-centered (G) mesh or Monkhorst-Pack (M) grid
1  1  1             ! subdivisions along the reciprocal lattice vectors
0  0  0             ! optional shift of the mesh
```

https://www.vasp.at/wiki/index.php/KPOINTS

VASP Inputs files – POTCAR

POTCAR contains the pseudopotential for each atomic species used in the calculation. Usually, we will not edit POTCARs. Below shows the beginning lines of POTCAR for O element.

```
PAW_PBE O 08Apr2002
  6.00000000000000
parameters from PSCTR are:
  VRHFIN =O: s2p4
  LEXCH  = PE
  EATOM  =   432.3788 eV,   31.7789 Ry

  TITEL  = PAW_PBE O 08Apr2002
  …
  POMASS =   16.000; ZVAL   =    6.000    mass and valenz
  …
  ENMAX  =  400.000; ENMIN  =  300.000 eV
```

If there are more than one element in your system, use ‘cat’ command to combine POTCARs.

```
$ cat ~/path/C/POTCAR ~/path/H/POTCAR ~/path/O/POTCAR > POTCAR
```

VASP submit script

```
#!/bin/bash

#SBATCH -A myallocation       # Allocation name
#SBATCH --nodes=1             # Total # of nodes
#SBATCH --ntasks=64     # Total # of MPI tasks
#SBATCH --time=00:30:00       # Total run time limit (hh:mm:ss)
#SBATCH -p wholenode           # Queue (partition) name

# Manage processing environment, load compilers and
applications.
module purge
module load gcc/11.2.0  openmpi/4.0.6
module load vasp/6.3.0 # or module load vasp/5.4.4.pl2
module list

# Launch MPI code
mpirun -np $SLURM_NTASKS vasp_std
```

Find allocation name with command, mybalance

```
x-     @login00.anvil:[~] $ mybalance

Allocation      Type    SU Limit    SU Usage   SU Usage  SU Balance
Account                             (account)    (user)
==============  ====  ==========  ========== ==========  ==========
asc170016        CPU    100000.0     54238.7      552.3     45761.3
asc170016-gpu    GPU      5000.0       144.3        0.0      4855.7
```

Find available partitions, showpartitions

```
x-      @login02.anvil:[~] $ showpartitions
Partition statistics for cluster anvil at T
     Partition     #Nodes     #CPU_cores  C
     Name State Total  Idle  Total   Idle F
wholenode    up   750     1  96000   2304
 standard    up   750     1  96000   2304
 shared:*    up   250   176  32000  23139
     wide    up   750     1  96000   2304
  highmem    up    32    28   4096   3633
    debug    up    17     1   2176    314
      gpu    up    16     1   2048   1684
gpu-debug    up    16     1   2048   1684
```

Anvil partitions

All Anvil nodes have 128 processor cores, 256 GB or 1 TB of RAM.

Anvil Production Queues						
Queue Name	**Node Type**	**Max Nodes per Job**	**Max Cores per Job**	**Max Duration**	**Max running Jobs in Queue**	**Max running + submitted Jobs in Queue**
debug	regular	2 nodes	256 cores	2 hrs	1	2
gpu-debug	gpu	1 node	2 gpus	0.5 hrs	1	2
wholenode	regular	16 nodes	2,048 cores	96 hrs	64	2500
wide	regular	56 nodes	7,168 cores	12 hrs	5	10
shared	regular	1 node	128 cores	96 hrs	6400 cores	-
highmem	large-memory	1 node	128 cores	48 hrs	2	4
gpu	gpu	-	-	48 hrs	-	-

https://www.rcac.purdue.edu/knowledge/anvil/run/partitions

How to charge SUs

- The charge unit for Anvil is the Service Unit (SU).
- 1 SU: use of 1 core (≤ 2GB memory) for 1 hour
- Example: a job running with 4 cores for 2 hours:

```
x-     @login00.anvil:[~] $ mybalance

Allocation     Type    SU Limit    SU Usage   SU Usage  SU Balance
Account                            (account)    (user)
=============  ====  ==========  ========== ==========  ==========
asc170016       CPU    100000.0     54238.7      552.3     45761.3
asc170016-gpu   GPU      5000.0       144.3        0.0      4855.7
```

Partitions	Charged SUs
shared, debug	If ~8GB memory, SU = 4 cores x 2 hours = 8
wholenode (standard), wide	SU = 128 cores x 2 hours = 256
highmem	SU = 4 x regular SU amount

- Command to check SU usage for one job

```
@login02.anvil:[~] $ /usr/local/bin/jobsu <jobid>
```

More information about SU charging:
https://www.rcac.purdue.edu/knowledge/anvil/run/accounting

A few useful commands

- sbatch <submit_script>
- scancel <jobid>
- squeue -ul username
- scontrol update job=<jobid> timelimit=03-00:00:00
- jobinfo <jobid>
- mybalance
- myquota

VASP on Anvil

Known errors users run into

Where to find error message?

- OUTCAR (in the end)

  normal termination:

```
          User time (sec):       24.400
        System time (sec):        1.070
       Elapsed time (sec):       26.153

  Maximum memory used (kb):      247100.
  Average memory used (kb):          N/A

        Minor page faults:       162455
        Major page faults:          107
Voluntary context switches:        1806
```

- the .out file (in the end) #SBATCH --output slurm.%N.%j.out

  normal termination:

```
 reached required accuracy - stopping structural energy minimisation
```

- the .err file (in the end) #SBATCH --error=slurm.%N.%j.err

VASP not found

Error:

module: command not found

Root cause:

- No access to VASP modules on Anvil
- check your unix groups with command, groups

```
@login02.anvil:[~] $ groups
x-acs00000 vasp6 vasp5
```

Solution:

send a ticket to ask for VASP access, in the ticket, we need your email address associated with a valid VASP license

https://support.access-ci.org/user/login?destination=/open-a-ticket

Out of memory

Error:

srun: error: axxx: task xx: Out Of Memory

Root cause:

- Not enough cores or space

  check the memory usage with 'jobinfo <jobid>'

  $ jobinfo 6364002
  …
  Mem reserved : xxxxM
  Max Mem used : xxxxM

Solutions:

- reduce the memory demand by reducing ENCUT or KPOINTS
- increase the number of cores
- use wholenode or highmem partition

not divisible by NPAR

Error:

M_divide: can not subdivide

Root cause:

- total number of cores is not divisible by NPAR

Solutions:

- Use default: NPAR default is the same as the total number of cores
- Set NPAR ~ sqrt (total number of cores)

Numa_num

Error:

node_info->numa_num <= ((MPIDI_SHMGR_SYNCPAGE_SIZE / MPIDI_SHMGR_FLAG_SPACE) - 1)

Root cause:

- Unknown
- Good guess: intel mpi on Anvil

Workarounds:

Add '--mpi=pmi2' to your srun or mpirun,

```
srun -n $SLURM_NTASKS --mpi=pmi2 …
```

Segmentation fault

This error may not link to one specific cause, possible reasons can be:

- insufficient memory: not enough cores or space at $HOME or $PROJECT
- POSCAR structure
- MPI abort

  If use intel mpi, try to add '--mpi=pmi2'

VASP on Anvil

Best practice

How many cores/nodes to use?

- VASP in general considered to scale up to 1 core/atom.
  - Conservatively, go with 0.5 core/atom or slightly less
  - If use dense kpoints, use 0.5 core/atom for each kpoint group; the total number of cores = KPAR x 0.5 core/atom x (# of atoms)
  - If there are multiple images, total number of cores = IMAGES x 0.5 core/atom x (# of atoms)
- Using larger number of cores may not reduce the time to solution cost effectively
  - VASP may spend most of the time in the communication even if it does not run into error

Choose right file system

```
x-     @login04.anvil:[~] $ myquota

Type     Location           Size    Limit    Use        Files    Limit     Use
==========================================================================
home     x-               983.5MB   25.0GB     4%            -        -      -
scratch  anvil              512KB  100.0TB  0.00%           0k   1,000k  0.00%
projects x-asc170016     358.0GB    5.0TB     7%          94k   1,048k     9%
```

– Your homes directory, $HOME, is not the right file system to run your VASP jobs – You may exceed your home quota, 25 GB and receive ‘out of memory’ error.

– The scratch directory, $SCRATCH, are recommended file systems to run your jobs both for a larger storage space and a better I/O performance. There are 100TB scratch quota for each user on Anvil.

Important note: your scratch folder will be purged regularly.

– Back up your important files to your project directory, /anvil/projects/x-000000, which has 5TB quota.

Short scaling tests

- Short scaling tests are recommended.
  - starting with 1 core/atom
  - Parameters to consider: NPAR/NCORE, KPAR

    NPAR ~ sqrt (total number of cores) performs better than NPAR=1 or the default NPAR

    KPAR ~ an integer divisor of the total number of cores (e.g. number of nodes)

# of nodes	# of cores	run time (T_r), sec	expected run time (T_{er}), sec	scaling efficiency = T_{er}* 100/T_r , %
0.25	32	228	228	228*100/228=100
0.5	64	125	228/2=114	114*100/125=91
1	128	96	114/2=57	57*100/96=59
2	256	112	57/2=29	29*100/112=26

- Use debug partition to see if your jobs could run to completion before submitting the long jobs

Anvil support

- Anvil user guide

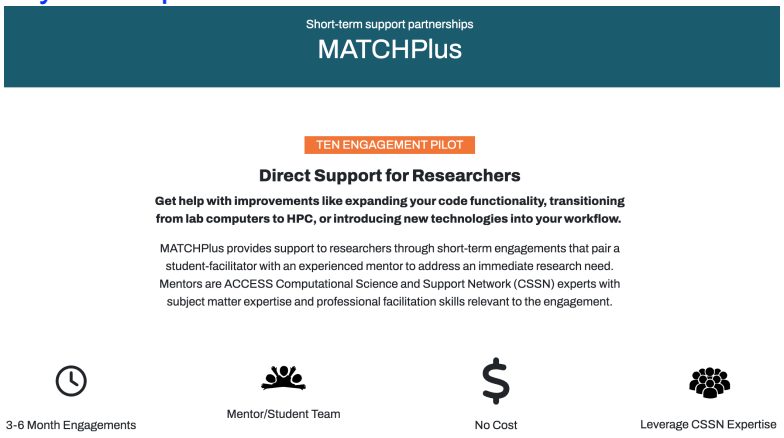
  https://www.rcac.purdue.edu/knowledge/anvil

- For general questions, submit a ticket to ACCESS Help Desk

  https://support.access-ci.org/user/login?destination=/open-a-ticket

- For complex consultation request, submit your request to MATCHPlus

  https://support.access-ci.org/matchplus



THANK YOU!