janosh / compile-vasp-m1.md

Last active last month

VASP M1 Mac Compilation Guide

compile-vasp-m1.md

Compile VASP on M1 Mac

> Courtesy of Alex Ganose @utf with additions from yours truly @janosh. Dated Mar 28, 2022.

1. Install Xcode command line tools

   ```
   xcode-select --install
   ```

2. Install gcc, OpenMPI and OpenMP using homebrew

   ```
   brew install gcc openmpi scalapack fftw qd openblas
   ```

   Consider appending `hdf5` if you want to compile VASP with HDF5 support.

3. Compile VASP

   These instructions are for VASP 6.3.0; they should be transferable to other versions of VASP but the variable names may be different

   ```
   cd /path/to/vasp-6.x.y
   cp arch/makefile.include.gnu_omp makefile.include
   ```

   Then edit `makefile.include` as follows:

   - Add the following to CPP_OPTIONS:

```
        -D_OPENMP \
        -Dqd_emulate
```

- Change all instances of `gcc` to `gcc-11` and `g++` to `g++-11`
- Add the following lines after `LLIBS        = -lstdc++`. This is necessary to emulate quad precision.

```
QD          ?= /opt/homebrew/
LLIBS       += -L$(QD)/lib -lqdmod -lqd
INCS        += -I$(QD)/include/qd
```

- Set `SCALAPACK_ROOT ?= /opt/homebrew`
- Set `OPENBLAS_ROOT ?= /opt/homebrew/Cellar/openblas/0.3.20` (Double check this is the path on your system)
- Set `FFTW_ROOT ?= /opt/homebrew`
- (optional but recommended by VASP) For HDF5 support, add

```
CPP_OPTIONS+= -DVASP_HDF5
HDF5_ROOT   ?= /opt/homebrew/
LLIBS       += -L$(HDF5_ROOT)/lib -lhdf5_fortran
INCS        += -I$(HDF5_ROOT)/include
```

4. Finally, run:

```
make all
```

If a previous compilation failed, remember to run `make veryclean` to start from a clean slate. Fixes `gfortran` errors like

> Fatal Error: `string.mod` not found

Resulting `makefile.include` with all modifications

See `makefile.include` below.

Benchmarking

Initial performance testing suggests optimal parameters for the M1 Pro chip with 8 performance, 2 efficiency cores and 16" MBP cooling are

```
export OMP_NUM_THREADS=1 # important
mpiexec -np 8 vasp_std
NCORE = 4 # in INCAR
```

	n_proc	n_threads	n_core	elapsed (sec)
0	1	1	2	93.3
1	1	1	4	92.8
2	1	2	2	82.8
3	1	2	4	82.7
4	2	1	2	42.8
5	2	1	4	42.9
6	2	2	2	52.9
7	2	2	4	52.7
8	4	1	2	32.9
9	4	1	4	32.9
10	4	2	2	52.9
11	4	2	4	53.0
12	8	1	2	32.8
13	8	1	4	22.8
14	8	2	2	62.8
15	8	2	4	62.9

Brings wall time for this Si2 relaxation down to 23 sec.

```python
from time import perf_counter

from atomate2.vasp.jobs.core import RelaxMaker
from atomate2.vasp.powerups import update_user_incar_settings
from jobflow import run_locally
from pymatgen.core import Structure


start = perf_counter()

# FCC silicon structure
si_structure = Structure(
    lattice=[[0, 2.73, 2.73], [2.73, 0, 2.73], [2.73, 2.73, 0]],
    species=["Si", "Si"],
    coords=[[0, 0, 0], [0.25, 0.25, 0.25]],
)

# relax job to optimize structure
relax_job = RelaxMaker().make(si_structure)

relax_job = update_user_incar_settings(relax_job, {"NCORE": 4})

# run job
run_locally(relax_job, create_folders=True, ensure_success=True)

print(f"Si relaxation took {perf_counter() - start:.3f} sec")
```

makefile.include

```
# Default precompiler options
CPP_OPTIONS = -DHOST=\"LinuxGNU\" \
              -DMPI -DMPI_BLOCK=8000 -Duse_collective \
              -DscaLAPACK \
              -DCACHE_SIZE=4000 \
              -Davoidalloc \
              -Dvasp6 \
              -Duse_bse_te \
              -Dtbdyn \
              -Dfock_dblbuf \
              -D_OPENMP \
              -Dqd_emulate

CPP         = gcc-11 -E -C -w $*$(FUFFIX) >$*$(SUFFIX) $(CPP_OPTIONS)

FC          = mpif90 -fopenmp
FCL         = mpif90 -fopenmp

FREE        = -ffree-form -ffree-line-length-none

FFLAGS      = -w -ffpe-summary=invalid,zero,overflow -L /opt/homebrew/Cellar/gcc/11.2.0_3/lib/gcc/

OFLAG       = -O2
OFLAG_IN    = $(OFLAG)
DEBUG       = -O0

OBJECTS     = fftmpiw.o fftmpi_map.o fftw3d.o fft3dlib.o
OBJECTS_O1 += fftw3d.o fftmpi.o fftmpiw.o
OBJECTS_O2 += fft3dlib.o

# For what used to be vasp.5.lib
CPP_LIB     = $(CPP)
FC_LIB      = $(FC)
CC_LIB      = gcc-11
CFLAGS_LIB  = -O
FFLAGS_LIB  = -O1
```

```
FREE_LIB    = $(FREE)

OBJECTS_LIB = linpack_double.o

# For the parser library
CXX_PARS    = g++-11
LLIBS        = -lstdc++
QD          ?= /opt/homebrew
LLIBS       += -L$(QD)/lib -lqdmod -lqd
INCS        += -I$(QD)/include/qd

##
## Customize as of this point! Of course you may change the preceding
## part of this file as well if you like, but it should rarely be
## necessary ...
##

# When compiling on the target machine itself, change this to the
# relevant target when cross-compiling for another architecture
FFLAGS     += -march=native

# For gcc-10 and higher (comment out for older versions)
FFLAGS     += -fallow-argument-mismatch

# BLAS and LAPACK (mandatory)
OPENBLAS_ROOT ?= /opt/homebrew/Cellar/openblas/0.3.20
BLASPACK    = -L$(OPENBLAS_ROOT)/lib -lopenblas

# scaLAPACK (mandatory)
SCALAPACK_ROOT ?= /opt/homebrew
SCALAPACK   = -L$(SCALAPACK_ROOT)/lib -lscalapack

LLIBS      += $(SCALAPACK) $(BLASPACK)

# FFTW (mandatory)
FFTW_ROOT  ?= /opt/homebrew
LLIBS      += -L$(FFTW_ROOT)/lib -lfftw3 -lfftw3_omp
INCS       += -I$(FFTW_ROOT)/include

# HDF5-support (optional but strongly recommended)
#CPP_OPTIONS+= -DVASP_HDF5
#HDF5_ROOT  ?= /path/to/your/hdf5/installation
#LLIBS      += -L$(HDF5_ROOT)/lib -lhdf5_fortran
#INCS       += -I$(HDF5_ROOT)/include

# For the VASP-2-Wannier90 interface (optional)
#CPP_OPTIONS    += -DVASP2WANNIER90
#WANNIER90_ROOT ?= /path/to/your/wannier90/installation
#LLIBS          += -L$(WANNIER90_ROOT)/lib -lwannier

# For the fftlib library (experimental)
#CPP_OPTIONS+= -Dsysv
#FCL        += fftlib.o
```

```
#CXX_FFTLIB  = g++-11 -fopenmp -std=c++11 -DFFTLIB_THREADSAFE
#INCS_FFTLIB = -I./include -I$(FFTW_ROOT)/include
#LIBS       += fftlib
#LLIBS      += -ldl
```

vasp-perf-grid-search.py

```python
"""This script grid-searches OMP_NUM_THREADS, NCORE and number of MPI processes for
minimal VASP runtime on a simple Si2 relaxation. It writes the results to CSV and copies
markdown table to clipboard. Requires Python 3.10. Invoke with

python vasp-perf-grid-search.py 2>&1 | tee Si-relax.log

to keep a log.
"""

import os
import warnings
from itertools import product
from time import perf_counter, sleep

import pandas as pd
from atomate2.vasp.jobs.core import RelaxMaker
from atomate2.vasp.powerups import update_user_incar_settings
from jobflow import run_locally
from pandas.io.clipboard import clipboard_set
from pymatgen.core import Structure


warnings.filterwarnings("ignore")  # suppress pymatgen warnings clogging up the logs

VASP_BIN = "/Users/janosh/dev/vasp/compiled/vasp_std_6.3.0_m1"
results: list[tuple[int, int, int, float]] = []

# construct an FCC silicon structure
si_structure = Structure(
    lattice=[[0, 2.73, 2.73], [2.73, 0, 2.73], [2.73, 2.73, 0]],
    species=["Si", "Si"],
    coords=[[0, 0, 0], [0.25, 0.25, 0.25]],
)

# grid-search OMP_NUM_THREADS, NCORE and number of MPI processes
try:
    prod = list(product([1, 2, 4, 8], [1, 2], [2, 4]))
    for idx, (n_proc, n_threads, n_core) in enumerate(prod, 1):

        os.environ["OMP_NUM_THREADS"] = str(n_threads)

        print(f"Run {idx} / {len(prod)}")

        # make a relax job to optimize the structure
        relax_job = RelaxMaker(
            run_vasp_kwargs={"vasp_cmd": f"mpiexec -np {n_proc} {VASP_BIN}"}
        ).make(si_structure)

        relax_job = update_user_incar_settings(relax_job, {"NCORE": n_core})

        start = perf_counter()
        # run the job
        run_locally(relax_job, create_folders=True, ensure_success=True)

        elapsed = perf_counter() - start
        print(
            f"Si relaxation with {n_proc=}, {n_threads=}, {n_core=} took "
            f"{elapsed:.1f} sec"
        )
        results.append((n_proc, n_threads, n_core, elapsed))

        print("Waiting 10 secs to cooldown...\n\n", flush=True)
        sleep(10)  # so every run is a bit more like the first


except KeyboardInterrupt:  # exit gracefully on ctrl+c and write partial results
    print("Job was interrupted")


df = pd.DataFrame(results, columns=["n_proc", "n_threads", "n_core", "elapsed"])
df.round(2).to_csv("vasp-perf-results.csv")
clipboard_set(df.to_markdown())
```

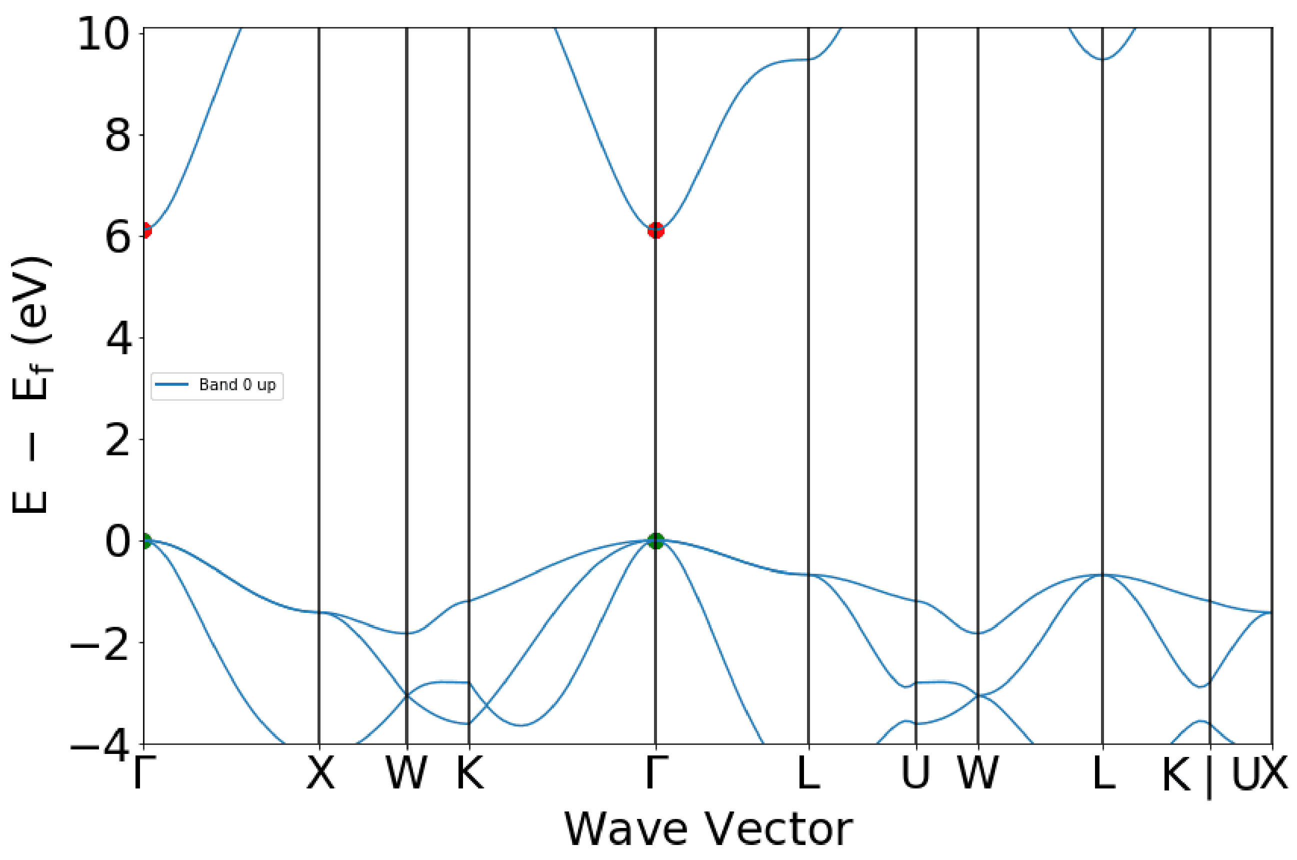
janosh commented on Apr 2, 2022 Author

Note to self: Don't get overconfident thinking M1 Pro could handle HSE bandstructure calcs.

This static MgO HSE calc took 15445.3 sec = 4.3 h.

```
import os
from time import perf_counter

from atomate2.vasp.flows.core import HSEBandStructureMaker
from jobflow import run_locally
from pymatgen.core import Structure


# construct a rock salt MgO structure
mgo_structure = Structure(
    lattice=[[0, 2.13, 2.13], [2.13, 0, 2.13], [2.13, 2.13, 0]],
    species=["Mg", "O"],
    coords=[[0, 0, 0], [0.5, 0.5, 0.5]],
)

# make a band structure flow to optimise the structure and obtain the band structure
bandstructure_flow = HSEBandStructureMaker().make(mgo_structure)

start = perf_counter()
# run the job
run_locally(bandstructure_flow, create_folders=True, ensure_success=True)
```

```
    elapsed = perf_counter() - start
    print(f"MgO HSE calc took {elapsed:.1f} sec")
```



zhuligs commented on Sep 21, 2022

Hello, thanks a lot for this.
I just followed your instructions. However, it failed at the last step when linking vasp.

ld: in parser/libparser.a(locproj.tab.h), archive member 'locproj.tab.h' with length 2696 is not mach-o or llvm bitcode file 'parser/libparser.a' for architecture arm64

Any suggestion? Thanks a lot.

janosh commented on Sep 21, 2022 Author

Sounds like the linker is trying to link an x86 object which doesn't exist on your arm64 machine. But haven't seen this error before so don't know how to fix. Are you sure you set `FFLAGS += -march=native` ? Maybe start the process from scratch to make sure you didn't miss a step.

zhuligs commented on Sep 21, 2022

Yes, I set `FFLAGS += -march=native` .
I tried to compile vasp on M1 Max, macOS 12.6 with gcc/gfortran-12 (homebrew). Not sure if M1 Max causes the problem.
Anyway, thanks very much! I'll try to figure it out.

zhuligs commented on Oct 4, 2022

update:
I changed line 16 of src/parser/makefile to
`ar vq libparser.a $(CPPOBJ_PARS) $(COBJ_PARS)`

And it worked for me.

HuangJiaLian commented on Nov 19, 2022 • edited

> Yes, I set `FFLAGS += -march=native` . I tried to compile vasp on M1 Max, macOS 12.6 with gcc/gfortran-12 (homebrew). Not sure if M1 Max causes the problem. Anyway, thanks very much! I'll try to figure it out.

Same problem on Intel 2015 Macbook Pro macOS 12.6 vasp.6.1.1

HuangJiaLian commented on Nov 19, 2022

> update: I changed line 16 of src/parser/makefile to `ar vq libparser.a $(CPPOBJ_PARS) $(COBJ_PARS)`
>
> And it worked for me.

I changed it, then `make all` but it seems no help. Does it need to run `make veryclean` after I change `src/parser/makefile` ? @zhuligs

ML5517 commented on Feb 1 • edited

Hello, thank you very much for posting it.
I just followed your instruction and set FFLAGS += -march=native, but it failed at the last step when linking vasp.

ld: warning: ignoring file /opt/homebrew/Cellar/openblas/0.3.21/lib/libopenblas.dylib, building for macOS-x86_64 but attempting to link with file built for macOS-arm64.

Do you have any idea how to fix it? Thank you very much!

ML5517 commented on Feb 5

> Hello, thank you very much for posting it. I just followed your instruction and set FFLAGS += -march=native, but it failed at the last step when linking vasp.
>
> ld: warning: ignoring file /opt/homebrew/Cellar/openblas/0.3.21/lib/libopenblas.dylib, building for macOS-x86_64 but attempting to link with file built for macOS-arm64.
>
> Do you have any idea how to fix it? Thank you very much!

I ran

```
env /usr/bin/arch -arm64 /bin/zsh --login
```

And then

```
make all arch=ARM64
```

And It works for me

lauren-walters commented on Mar 23

Thank you so much for this post! It was easy to follow and with a few modifications to the makefile.include (below) I was able to build a vasp_std for my m2 mac with some newer versions of gcc and openblas. However, I when testing vasp, each node is running vasp simultaneously. I've seen other posts about this from years back, but a lot seem to tell you to go talk to your hpc system administrator. I'm assuming I didn't link a version of mpi that runs well, but am not totally sure where to go from here. Any help is appreciated!

```
# Default precompiler options
CPP_OPTIONS = -DHOST=\"LinuxGNU\" \
              -DMPI -DMPI_BLOCK=8000 -Duse_collective \
              -DscaLAPACK \
              -DCACHE_SIZE=4000 \
              -Davoidalloc \
              -Dvasp6 \
              -Duse_bse_te \
              -Dtbdyn \
              -Dfock_dblbuf \
              -D_OPENMP \
              -Dqd_emulate


CPP         = gcc-12 -E -C -w $*$(FUFFIX) >$*$(SUFFIX) $(CPP_OPTIONS)

FC          = /opt/homebrew/bin/mpif90 -fopenmp
FCL         = /opt/homebrew/bin/mpif90 -fopenmp

FREE        = -ffree-form -ffree-line-length-none

FFLAGS      = -w -ffpe-summary=invalid,zero,overflow -L /opt/homebrew/Cellar/gcc/12.2.0/lib/gcc/12

OFLAG       = -O2
OFLAG_IN    = $(OFLAG)
DEBUG       = -O0

OBJECTS     = fftmpiw.o fftmpi_map.o fftw3d.o fft3dlib.o
OBJECTS_O1 += fftw3d.o fftmpi.o fftmpiw.o
OBJECTS_O2 += fft3dlib.o

# For what used to be vasp.5.lib
CPP_LIB     = $(CPP)
FC_LIB      = $(FC)
CC_LIB      = gcc-12
CFLAGS_LIB  = -O
```

```
FFLAGS_LIB  = -O1
FREE_LIB    = $(FREE)

OBJECTS_LIB = linpack_double.o

# For the parser library
CXX_PARS    = g++-12
LLIBS       = -lstdc++
QD         ?= /opt/homebrew
LLIBS      += -L$(QD)/lib -lqdmod -lqd
INCS       += -I$(QD)/include/qd

##
## Customize as of this point! Of course you may change the preceding
## part of this file as well if you like, but it should rarely be
## necessary ...
##

# When compiling on the target machine itself, change this to the
# relevant target when cross-compiling for another architecture
FFLAGS     += -march=native

# For gcc-10 and higher (comment out for older versions)
FFLAGS     += -fallow-argument-mismatch

# BLAS and LAPACK (mandatory)
OPENBLAS_ROOT ?= /opt/homebrew/Cellar/openblas/0.3.21
BLASPACK    = -L$(OPENBLAS_ROOT)/lib -lopenblas

# scaLAPACK (mandatory)
SCALAPACK_ROOT ?= /opt/homebrew
SCALAPACK   = -L$(SCALAPACK_ROOT)/lib -lscalapack

LLIBS      += $(SCALAPACK) $(BLASPACK)

# FFTW (mandatory)
FFTW_ROOT  ?= /opt/homebrew
LLIBS      += -L$(FFTW_ROOT)/lib -lfftw3 -lfftw3_omp
INCS       += -I$(FFTW_ROOT)/include

# HDF5-support (optional but strongly recommended)
#CPP_OPTIONS+= -DVASP_HDF5
#HDF5_ROOT  ?= /path/to/your/hdf5/installation
#LLIBS      += -L$(HDF5_ROOT)/lib -lhdf5_fortran
#INCS       += -I$(HDF5_ROOT)/include

# For the VASP-2-Wannier90 interface (optional)
#CPP_OPTIONS    += -DVASP2WANNIER90
#WANNIER90_ROOT ?= /path/to/your/wannier90/installation
#LLIBS          += -L$(WANNIER90_ROOT)/lib -lwannier

# For the fftlib library (experimental)
#CPP_OPTIONS+= -Dsysv
#FCL        += `fftlib.o`
#CXX_FFTLIB  = g++-12 -fopenmp -std=c++12 -DFFTLIB_THREADSAFE
#INCS_FFTLIB = -I./include -I$(FFTW_ROOT)/include
#LIBS       += fftlib
#LLIBS      += -ldl
```

hrushikesh-s commented on May 12

Thanks @janosh for this step-by-step explanation of compiling VASP on M1 mac. I was able to successfully compile vasp.6.4.1 on my mac with M1 pro chip.

tpcklaver commented on Jul 4 • edited

Thanks @janosh for these instructions on my M2 Pro Macbook, running macOS Ventura. I tried using them to compile VASP 5.4.4. I made just two tiny tweaks to your makefile.include for newer versions brew gave me for gcc and openblas, i.e. gcc-11 -> gcc-13 and openblas 0.3.20 -> 0.3.23. Things go ok up to the link stage, but then I get an Undefined symbols for architecture arm64 error, see command line output below. Have you (or anyone else compiling VASP on Apple Silicon) ever come across these errors, and would you have any idea how to fix them?

Kind regards,
Peter

```
mpif90 -fopenmp -o vasp c2f_interface.o base.o profiling.o openmp.o mpi.o mpi_shmem.o smart_allocate.o
xml.o constant.o jacobi.o main_mpi.o scala.o asa.o lattice.o poscar.o ini.o mgrid.o xclib.o vdw_nl.o
xclib_grad.o radial.o pseudo.o gridq.o ebs.o mkpoints.o wave.o wave_mpi.o wave_high.o bext.o spinsym.o
symlib.o symmetry.o lattlib.o random.o nonl.o nonlr.o nonl_high.o dfast.o choleski2.o mix.o hamil.o xcgrad.o
xcspin.o potex1.o potex2.o constrmag.o cl_shift.o relativistic.o LDApU.o paw_base.o metagga.o egrad.o
pawsym.o pawfock.o pawlhf.o rhfatm.o hyperfine.o paw.o mkpoints_full.o charge.o Lebedev-Laikov.o
stockholder.o dipol.o solvation.o pot.o dos.o elf.o tet.o tetweight.o hamil_rot.o chain.o dyna.o k-proj.o
sphpro.o us.o core_rel.o aedens.o wavpre.o wavpre_noio.o broyden.o dynbr.o reader.o writer.o tutor.o
xml_writer.o brent.o stufak.o fileio.o opergrid.o stepver.o chgloc.o fast_aug.o fock_multipole.o fock.o
fock_dbl.o mkpoints_change.o subrot_cluster.o sym_grad.o mymath.o npt_dynamics.o subdftd3.o internals.o
dynconstr.o dimer_heyden.o dvvtrajectory.o vdwforcefield.o hamil_high.o nmr.o pead.o subrot.o subrot_scf.o
paircorrection.o rpa_force.o force.o pwlhf.o gw_model.o optreal.o steep.o rmm-diis.o davidson.o
david_inner.o lcao_bare.o locproj.o electron.o rot.o electron_all.o shm.o pardens.o optics.o constr_cell_relax.o
stm.o finite_diff.o elpol.o hamil_lr.o rmm-diis_lr.o subrot_lr.o lr_helper.o hamil_lrf.o elinear_response.o
ilinear_response.o linear_optics.o setlocalpp.o wannier.o electron_OEP.o electron_lhf.o twoelectron4o.o
gauss_quad.o m_unirnk.o varpro.o minimax.o mlwf.o wnpr.o ratpol.o pade_fit.o screened_2e.o wave_cacher.o
crpa.o chi_base.o wpot.o local_field.o ump2.o ump2kpar.o fcidump.o ump2no.o bse_te.o bse.o acfdt.o chi.o
sydmat.o rmm-diis_mlr.o linear_response_NMR.o wannier_interpol.o linear_response.o dmft.o auger.o
dmatrix.o elphon.o fftmpiw.o fftmpi_map.o fftw3d.o fft3dlib.o main.o -Llib -ldmy -lstdc++ -
L/opt/homebrew/lib -lqdmod -lqd -L/opt/homebrew/lib -lscalapack -
L/opt/homebrew/Cellar/openblas/0.3.23/lib -lopenblas -L/opt/homebrew/lib -lfftw3 -lfftw3_omp
Undefined symbols for architecture arm64:
"_attachshmem_C", referenced from:
___mpi_shmem_MOD_shmem_alloc_r_3d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_r_2d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_r_1d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_c_3d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_c_2d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_c_1d in mpi_shmem.o
"_destroyshmem_C", referenced from:
___mpi_shmem_MOD_shmem_alloc_r_3d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_r_2d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_r_1d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_c_3d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_c_2d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_c_1d in mpi_shmem.o
"_detachshmem_C", referenced from:
___mpi_shmem_MOD_shmem_dealloc_r_3d in mpi_shmem.o
___mpi_shmem_MOD_shmem_dealloc_r_2d in mpi_shmem.o
___mpi_shmem_MOD_shmem_dealloc_r_1d in mpi_shmem.o
___mpi_shmem_MOD_shmem_dealloc_c_3d in mpi_shmem.o
___mpi_shmem_MOD_shmem_dealloc_c_2d in mpi_shmem.o
___mpi_shmem_MOD_shmem_dealloc_c_1d in mpi_shmem.o
"_fill_basis_info_C", referenced from:
___locproj_MOD_lprj_reader in locproj.o
"_free_parser_C", referenced fro "_free_parser_C", referenced fro "_free_parser_C", referenced from:
___locproj_MOD_lprj_reader in locproj.o
"_getshmem_C", referenced from:
___mpi_shmem_MOD_shmem_alloc_r_3d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_r_2d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_r_1d in mpi_shmem.o
```

```
___mpi_shmem_MOD_shmem_alloc_c_3d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_c_2d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_c_1d in mpi_shmem.o
"_getshmem_error_C", referenced from:
___mpi_shmem_MOD_shmem_alloc_r_3d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_r_2d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_r_1d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_c_3d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_c_2d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_c_1d in mpi_shmem.o
"_parse_file_C", referenced from:
___locproj_MOD_lprj_reader in locproj.o
ld: symbol(s) not found for architecture arm64
collect2: error: ld returned 1 exit status
make[2]: *** [vasp] Error 1
cp: vasp: No such file or directory
make[1]: *** [all] Error 1
make: *** [std] Error 2
bash-3.2$
bash-3.2$
```

hungpham2017 commented on Aug 26

> Thanks @janosh for these instructions on my M2 Pro Macbook, running macOS Ventura. I tried using them to compile VASP 5.4.4. I made just two tiny tweaks to your makefile.include for newer versions brew gave me for gcc and openblas, i.e. gcc-11 -> gcc-13 and openblas 0.3.20 -> 0.3.23. Things go ok up to the link stage, but then I get an Undefined symbols for architecture arm64 error, see command line output below. Have you (or anyone else compiling VASP on Apple Silicon) ever come across these errors, and would you have any idea how to fix them?
>
> Kind regards, Peter

```
mpif90 -fopenmp -o vasp c2f_interface.o base.o profiling.o openmp.o mpi.o mpi_shmem.o
smart_allocate.o xml.o constant.o jacobi.o main_mpi.o scala.o asa.o lattice.o poscar.o ini.o mgrid.o
xclib.o vdw_nl.o xclib_grad.o radial.o pseudo.o gridq.o ebs.o mkpoints.o wave.o wave_mpi.o
wave_high.o bext.o spinsym.o symlib.o symmetry.o lattlib.o random.o nonl.o nonlr.o nonl_high.o dfast.o
choleski2.o mix.o hamil.o xcgrad.o xcspin.o potex1.o potex2.o constrmag.o cl_shift.o relativistic.o
LDApU.o paw_base.o metagga.o egrad.o pawsym.o pawfock.o pawlhf.o rhfatm.o hyperfine.o paw.o
mkpoints_full.o charge.o Lebedev-Laikov.o stockholder.o dipol.o solvation.o pot.o dos.o elf.o tet.o
tetweight.o hamil_rot.o chain.o dyna.o k-proj.o sphpro.o us.o core_rel.o aedens.o wavpre.o
wavpre_noio.o broyden.o dynbr.o reader.o writer.o tutor.o xml_writer.o brent.o stufak.o fileio.o
opergrid.o stepver.o chgloc.o fast_aug.o fock_multipole.o fock.o fock_dbl.o mkpoints_change.o
subrot_cluster.o sym_grad.o mymath.o npt_dynamics.o subdftd3.o internals.o dynconstr.o
dimer_heyden.o dvvtrajectory.o vdwforcefield.o hamil_high.o nmr.o pead.o subrot.o subrot_scf.o
paircorrection.o rpa_force.o force.o pwlhf.o gw_model.o optreal.o steep.o rmm-diis.o davidson.o
david_inner.o lcao_bare.o locproj.o electron.o rot.o electron_all.o shm.o pardens.o optics.o
constr_cell_relax.o stm.o finite_diff.o elpol.o hamil_lr.o rmm-diis_lr.o subrot_lr.o lr_helper.o hamil_lrf.o
elinear_response.o ilinear_response.o linear_optics.o setlocalpp.o wannier.o electron_OEP.o electron_lhf.o
twoelectron4o.o gauss_quad.o m_unirnk.o varpro.o minimax.o mlwf.o wnpr.o ratpol.o pade_fit.o
screened_2e.o wave_cacher.o crpa.o chi_base.o wpot.o local_field.o ump2.o ump2kpar.o fcidump.o
ump2no.o bse_te.o bse.o acfdt.o chi.o sydmat.o rmm-diis_mlr.o linear_response_NMR.o
wannier_interpol.o linear_response.o dmft.o auger.o dmatrix.o elphon.o fftmpiw.o fftmpi_map.o fftw3d.o
fft3dlib.o main.o -Llib -ldmy -lstdc++ -L/opt/homebrew/lib -lqdmod -lqd -L/opt/homebrew/lib -
lscalapack -L/opt/homebrew/Cellar/openblas/0.3.23/lib -lopenblas -L/opt/homebrew/lib -lfftw3 -
lfftw3_omp Undefined symbols for architecture arm64: "_attachshmem_C", referenced from:
___mpi_shmem_MOD_shmem_alloc_r_3d in mpi_shmem.o ___mpi_shmem_MOD_shmem_alloc_r_2d in
mpi_shmem.o ___mpi_shmem_MOD_shmem_alloc_r_1d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_c_3d in mpi_shmem.o ___mpi_shmem_MOD_shmem_alloc_c_2d in
mpi_shmem.o ___mpi_shmem_MOD_shmem_alloc_c_1d in mpi_shmem.o "_destroyshmem_C",
referenced from: ___mpi_shmem_MOD_shmem_alloc_r_3d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_r_2d in mpi_shmem.o ___mpi_shmem_MOD_shmem_alloc_r_1d in
mpi_shmem.o ___mpi_shmem_MOD_shmem_alloc_c_3d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_c_2d in mpi_shmem.o ___mpi_shmem_MOD_shmem_alloc_c_1d in
mpi_shmem.o "_detachshmem_C", referenced from: ___mpi_shmem_MOD_shmem_dealloc_r_3d in
mpi_shmem.o ___mpi_shmem_MOD_shmem_dealloc_r_2d in mpi_shmem.o
___mpi_shmem_MOD_shmem_dealloc_r_1d in mpi_shmem.o ___mpi_shmem_MOD_shmem_dealloc_c_3d
in mpi_shmem.o ___mpi_shmem_MOD_shmem_dealloc_c_2d in mpi_shmem.o
___mpi_shmem_MOD_shmem_dealloc_c_1d in mpi_shmem.o "_fill_basis_info_C", referenced from:
___locproj_MOD_lprj_reader in locproj.o "_free_parser_C", referenced fro "_free_parser_C", referenced fro
"_free_parser_C", referenced from: ___locproj_MOD_lprj_reader in locproj.o "_getshmem_C", referenced
from: ___mpi_shmem_MOD_shmem_alloc_r_3d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_r_2d in mpi_shmem.o ___mpi_shmem_MOD_shmem_alloc_r_1d in
mpi_shmem.o ___mpi_shmem_MOD_shmem_alloc_c_3d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_c_2d in mpi_shmem.o ___mpi_shmem_MOD_shmem_alloc_c_1d in
mpi_shmem.o "_getshmem_error_C", referenced from: ___mpi_shmem_MOD_shmem_alloc_r_3d in
mpi_shmem.o ___mpi_shmem_MOD_shmem_alloc_r_2d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_r_1d in mpi_shmem.o ___mpi_shmem_MOD_shmem_alloc_c_3d in
mpi_shmem.o ___mpi_shmem_MOD_shmem_alloc_c_2d in mpi_shmem.o
___mpi_shmem_MOD_shmem_alloc_c_1d in mpi_shmem.o "_parse_file_C", referenced from:
___locproj_MOD_lprj_reader in locproj.o ld: symbol(s) not found for architecture arm64 collect2: error: ld
returned 1 exit status make[2]: *** [vasp] Error 1 cp: vasp: No such file or directory make[1]: *** [all] Error
1 make: *** [std] Error 2 bash-3.2$ bash-3.2$
```

Having this same problem on my M2 Mac, appreciate any help!

cinnabar0321 commented on Aug 28

I have this error showing up all the time "make libparser.a
make[1]: *** No rule to make target `sites.o', needed by` libparser.a'. Stop.
make: *** [all] Error 2"

My version of gcc is 13.0.1, so instead of making gcc-11, I used gcc-13. The path in the makefile.include was also changed accordingly. However, the returned error is the same. Can anyone please help? Thank you!

pavlvs-pinto commented last month

I've been trying to compile vasp6.4.2 on m2 mac, but I get this error message:
ld: unknown options: -commons
I used the makefile.include as posted by **@lauren-walters** with gcc-13 and g++-13. Perhaps someone also got this error and found some fix?
Many thanks in advance!