软件资源

本章只列出部分软件资源，完整的软件资源需要在集群上通过访问：

/fs00/software

/fs00/software/singularity-images/

目录以查询。如您需要的软件没有在这里列出，您可以自行登录集群查看，或通过服务群联系询问。

GNU
Intel
NVIDIA
AMD

Python
Golang
Open MPI
Tcl/TK
计算软件
VASP6 GPU 编译
Quantum Espresso
NAMD
OOMMF
Mumax3
支撑软件
Singularity Image
gnuplot
OVITO
Vim
Zsh
AlphaFold 2

- AlphaFold 3
- RoseTTAFold
- ParallelFold 2

GNU

GNU Compiler Collection

Name	Path	Module
GNU Compiler Collection (GCC) 12.1.0 & GNU Binutils 2.38	/fs00/software/gcc/12.1.0	gcc/12.1.0
GNU Compiler Collection (GCC) 11.3.0 & GNU Binutils 2.36.1	/fs00/software/gcc/11.3.0	gcc/11.3.0
GNU Compiler Collection (GCC) 10.5.0 & GNU Binutils 2.34	/fs00/software/gcc/10.5.0	gcc/10.5.0
GNU Compiler Collection (GCC) 9.5.0 & GNU Binutils 2.32	/fs00/software/gcc/9.5.0	gcc/9.5.0
GNU Compiler Collection (GCC) 8.5.0 & GNU Binutils 2.30	/fs00/software/gcc/8.5.0	gcc/8.5.0
GNU Compiler Collection (GCC) 7.5.0 & GNU Binutils 2.28.1	/fs00/software/gcc/7.5.0	gcc/7.5.0
GNU Compiler Collection (GCC) 6.5.0 & GNU Binutils 2.26.1	/fs00/software/gcc/6.5.0	gcc/6.5.0
GNU Compiler Collection (GCC) 5.4.0	/fs00/software/gcc/5.4.0	gcc/5.4.0

GNU Make

Name	Path	Module
GNU Make 4.3	/fs00/software/make/4.3	make/4.3
GNU Make 4.2.1	/fs00/software/make/4.2.1	make/4.2.1
GNU Make 4.2	/fs00/software/make/4.2	make/4.2

GNU Scientific Library

Name	Compiler	Path	Module
GNU Scientific Library (GSL) 2.7.1	GCC 12.1.0	/fs00/software/gsl/2.7.1-gcc12.1.0	gsl/2.7.1-gcc12.1.0

Name	Compiler	Path	Module
GNU Scientific Library (GSL) 2.5	GCC 8.3.0	/fs00/software/gsl/2.5-gcc8.3	gsl/2.5-gcc8.3

GNU C Library

Name	Compiler	Path	Module
GNU C Library (glibc) 2.36	GCC 12.1.0	/fs00/software/glibc/2.36-gcc12.1.0	glibc/2.36-gcc12.1.0
GNU C Library (glibc) 2.30	GCC 9.2.0	/fs00/software/glibc/2.30-gcc9.2.0	glibc/2.30-gcc9.2.0

GNU Binutils

Name	Compiler	Path	Module
GNU Binutils 2.38	GCC 12.1.0	/fs00/software/binutils/2.38-gcc12.1.0	binutils/2.38-gcc12.1.0
GNU Binutils 2.27	GCC 5.4.0	/fs00/software/binutils/2.27-gcc5.4.0	binutils/2.27-gcc5.4.0

Intel

Intel oneAPI

Name	Path	MODULEPATH
Intel oneAPI Base Toolkit 2024.0.1 Intel HPC Toolkit 2024.0.1	/fs00/software/intel/oneapi2024.0	/fs00/software/modulefiles/oneapi/2024.0

Intel Parallel Studio

Name	Path	Module
Intel Parallel Studio XE 2020 Update 2 Cluster Edition	/fs00/software/intel/ps2020u2	ips/2020u2
Intel Parallel Studio XE 2019 Update 5 Cluster Edition	/fs00/software/intel/ps2019u5	ips/2019u5
Intel Parallel Studio XE 2018 Update 4 Cluster Edition	/fs00/software/intel/ps2018u4	ips/2018u4
Intel Parallel Studio XE 2017 Update 6 Cluster Edition	/fs00/software/intel/ps2017u6	ips/2017u6
Intel Parallel Studio XE 2017 Update 2 Cluster Edition	/fs00/software/intel/ps2017u2	ips/2017u2
Intel Parallel Studio XE 2016 Update 4 Cluster Edition	/fs00/software/intel/ps2016u4	ips/2016u4
Intel Parallel Studio XE 2015 Update 6 Cluster Edition	/fs00/software/intel/ps2015u6	ips/2015u6
Intel Cluster Studio XE 2013 Service Pack 1 (SP1) Update 1	/fs00/software/intel/cs2013sp1u1	ics/2013sp1u1
Intel Cluster Studio XE 2013	/fs00/software/intel/cs2013	ics/2013
Intel Parallel Studio XE 2011 SP1 Update 3	/fs00/software/intel/ps2011sp1u3	ips/2011sp1u3

Intel Distribution for Python

Name	Path
Intel Distribution for Python 2.7 2019 Update 5	/fs00/software/intel/ps2019u5/intelpython2

Name	Path
Intel Distribution for Python 3.6 2019 Update 5	/fs00/software/intel/ps2019u5/intelpython3
Intel Distribution for Python 2.7 2018 Update 3	/fs00/software/intel/python2018u3/intelpython2
Intel Distribution for Python 3.6 2018 Update 3	/fs00/software/intel/python2018u3/intelpython3
Intel Distribution for Python 2.7 2017 Update 3	/fs00/software/intel/python2017u3/intelpython2
Intel Distribution for Python 3.5 2017 Update 3	/fs00/software/intel/python2017u3/intelpython3

NVIDIA

CUDA Toolkit

Name	Path	Module
CUDA Toolkit 12.3.1	/fs00/software/cuda/12.3.1	cuda/12.3.1
CUDA Toolkit 12.0.0	/fs00/software/cuda/12.0.0	cuda/12.0.0
CUDA Toolkit 11.8.0	/fs00/software/cuda/11.8.0	cuda/11.8.0
CUDA Toolkit 11.2.0	/fs00/software/cuda/11.2.0	cuda/11.2.0
CUDA Toolkit 10.2.89	/fs00/software/cuda/10.2.89	cuda/10.2.89
CUDA Toolkit 10.1.243	/fs00/software/cuda/10.1.243	cuda/10.1.243
CUDA Toolkit 10.0.130	/fs00/software/cuda/10.0.130	cuda/10.0.130
CUDA Toolkit 9.2.148	/fs00/software/cuda/9.2.148	cuda/9.2.148
CUDA Toolkit 9.0.176 with Patch 3	/fs00/software/cuda/9.0.176	cuda/9.0.176
CUDA Toolkit 8.0 GA2 8.0.61 with Patch 2	/fs00/software/cuda/8.0.61	cuda/8.0.61

cuDNN

Name	CUDA	Path	Module
cuDNN v8.9.7.29	12.x	/fs00/software/cudnn/8.9.7.29-cuda12	cudnn/8.9.7.29-cuda12
cuDNN v8.9.7.29	11.x	/fs00/software/cudnn/8.9.7.29-cuda11	cudnn/8.9.7.29-cuda11
cuDNN v8.7.0.84	11.x	/fs00/software/cudnn/8.7.0.84-cuda11	cudnn/8.7.0.84-cuda11
cuDNN v8.7.0.84	10.2	/fs00/software/cudnn/8.7.0.84-cuda10	cudnn/8.7.0.84-cuda10
cuDNN v8.1.1.33	11.2	/fs00/software/cudnn/11.2-v8.1.1.33	cudnn/11.2-v8.1.1.33
cuDNN v8.2.2.26	10.2	/fs00/software/cudnn/10.2-v8.2.2.26	cudnn/10.2-v8.2.2.26

Name	CUDA	Path	Module
cuDNN v7.6.5.32	10.2	/fs00/software/cudnn/10.2-v7.6.5.32	cudnn/10.2-v7.6.5.32
cuDNN v7.6.4.38	10.1	/fs00/software/cudnn/10.1-v7.6.4.38	cudnn/10.1-v7.6.4.38
cuDNN v7.6.5.32	10.0	/fs00/software/cudnn/10.0-v7.6.5.32	cudnn/10.0-v7.6.5.32
cuDNN v7.1.4	9.2	/fs00/software/cudnn/9.2-v7.1.4	cudnn/9.2-v7.1.4
cuDNN v7.1.4	9.0	/fs00/software/cudnn/9.0-v7.1.4	cudnn/9.0-v7.1.4
cuDNN v7.0.5	8.0	/fs00/software/cudnn/8.0-v7.0.5	cudnn/8.0-v7.0.5
cuDNN v6.0	8.0	/fs00/software/cudnn/8.0-v6.0	cudnn/8.0-v6.0
cuDNN v5.1	8.0	/fs00/software/cudnn/8.0-v5.1	cudnn/8.0-v5.1

HPC SDK

Name	Path	MODULEPATH
HPC SDK 23.11	/fs00/software/nvhpc/23.11	/fs00/software/nvhpc/23.11/modulefiles
HPC SDK 22.11	/fs00/software/nvhpc/22.11	/fs00/software/nvhpc/22.11/modulefiles
HPC SDK 21.3	/fs00/software/nvhpc/21.3	/fs00/software/nvhpc/21.3/modulefiles
HPC SDK 20.9	/fs00/software/nvhpc/20.9	/fs00/software/nvhpc/20.9/modulefiles

HPC-X

Name	CUDA	Path	MODULEPATH
HPC-X 2.17.1	12.x	/fs00/software/hpcx/2.17.1-cuda12	/fs00/software/hpcx/2.17.1-cuda12/modulefiles

NCCL

Name	CUDA	Path	Module

NCCL 2.19.3	12.3	/fs00/software/nccl/2.19.3-cuda12.3	nccl/2.19.3-cuda12.3
NCCL 2.16.2	12.0	/fs00/software/nccl/2.16.2-cuda12.0	nccl/2.16.2-cuda12.0
NCCL 2.16.2	11.8	/fs00/software/nccl/2.16.2-cuda11.8	nccl/2.16.2-cuda11.8
NCCL 2.16.2	11.0	/fs00/software/nccl/2.16.2-cuda11.0	nccl/2.16.2-cuda11.0
NCCL v2.5.6	10.2	/fs00/software/nccl/10.2-v2.5.6	nccl/10.2-v2.5.6
NCCL v2.4.8	10.1	/fs00/software/nccl/10.1-v2.4.8	nccl/10.1-v2.4.8

TensorRT

Name	CUDA	cuDNN	Path	Module
TensorRT 8.6.1.6	12.0		/fs00/software/tensorrt/8.6.1.6-cuda12.0	tensorrt/8.6.1.6-cuda12.0
TensorRT 8.6.1.6	11.8		/fs00/software/tensorrt/8.6.1.6-cuda11.8	tensorrt/8.6.1.6-cuda11.8
TensorRT 8.5.2.2	11.8	8.6	/fs00/software/tensorrt/8.5.2.2-cuda11.8-cudnn8.6	tensorrt/8.5.2.2-cuda11.8-cudnn8.6
TensorRT 8.5.2.2	10.2	8.6	/fs00/software/tensorrt/8.5.2.2-cuda10.2-cudnn8.6	tensorrt/8.5.2.2-cuda10.2-cudnn8.6
TensorRT 8.2.0.6	11.4	8.2	/fs00/software/tensorrt/8.2.0.6-cuda11.4-cudnn8.2	tensorrt/8.2.0.6-cuda11.4-cudnn8.2
TensorRT 8.2.0.6	10.2	8.2	/fs00/software/tensorrt/8.2.0.6-cuda11.4-cudnn8.2	tensorrt/8.2.0.6-cuda10.2-cudnn8.2

AMD

AMD Optimizing C/C++ Compiler

Name	Path	Module
AMD Optimizing C/C++ Compiler 2.3.0 (AOCC)	/fs00/software/aocc/2.3.0	aocc/2.3.0
AMD Optimizing C/C++ Compiler 2.1.0 (AOCC)	/fs00/software/aocc/2.1.0	aocc/2.1.0
AMD Optimizing C/C++ Compiler 2.0.0 (AOCC)	/fs00/software/aocc/2.0.0	aocc/2.0.0

AMD Optimizing CPU Libraries

Name	Path	Module
AMD Optimizing CPU Libraries 2.2 (AOCL)	/fs00/software/aocl/2.2	aocl/2.2
AMD Optimizing CPU Libraries 2.0 (AOCL)	/fs00/software/aocl/2.0	aocl/2.0

Python

□□□□□□□□ **License**□□□□□□□□□□

Anaconda

Name	Path	Module
Anaconda 3 (Python3) Latest	/fs00/software/anaconda/3	anaconda/3
Anaconda 2 (Python2) Latest	/fs00/software/anaconda/2	anaconda/2
Anaconda 5.0.1 (Python 3.6)	/fs00/software/anaconda/3-5.0.1	anaconda/3-5.0.1
Anaconda 5.0.1 (Python 2.7)	/fs00/software/anaconda/2-5.0.1	anaconda/2-5.0.1
Anaconda 3.4.1 (Python 3.6)	/fs00/software/anaconda/3-3.4.1	anaconda/3-3.4.1
Anaconda 3.4.1 (Python 2.7)	/fs00/software/anaconda/2-3.4.1	anaconda/2-3.4.1

Golang

Golang

Name	Path	Module
Golang 1.21.6	/fs00/software/golang/1.21.6	golang/1.21.6
Golang 1.19.5	/fs00/software/golang/1.19.5	golang/1.19.5
Golang 1.18.10	/fs00/software/golang/1.18.7	golang/1.18.10
Golang 1.17.13	/fs00/software/golang/1.17.13	golang/1.17.13
Golang 1.16.15	/fs00/software/golang/1.16.15	golang/1.16.15
Golang 1.15.15	/fs00/software/golang/1.15.15	golang/1.15.15

Open MPI

Open MPI

Name	Compiler	Path	Module
Open MPI 4.1.2	GNU Compiler Collection (GCC) 11.2.0	/fs00/software/openmpi/4.1.2-gcc11.2.0	openmpi/4.1.2-gcc11.2.0
Open MPI 3.1.2	GNU Compiler Collection (GCC) 8.2.0	/fs00/software/openmpi/3.1.2-gcc8.2.0	openmpi/3.1.2-gcc8.2.0
Open MPI 1.10.0	Intel C++ Compiler XE 15.0 Update 3 & Fortran Compiler XE 15.0 Update 3	/fs00/software/openmpi/1.10.0-iccifort-15.0.3	openmpi/1.10.0-iccifort-15.0.3
Open MPI 1.10.0	GNU Compiler Collection (GCC) 5.2.0	/fs00/software/openmpi/1.10.0-gcc-5.2.0	openmpi/1.10.0-gcc-5.2.0
Open MPI 1.10.5	GNU Compiler Collection (GCC) 5.4.0	/fs00/software/openmpi/1.10.5-gcc5.4.0	openmpi/1.10.5-gcc5.4.0

Tcl/TK

Tcl/Tk

Name	Compiler	Path	Module
Tcl/Tk 8.6.12	GNU Compiler Collection (GCC) 11.2.0	/fs00/software/tcl/8.6.12-gcc11.2.0	tcl/8.6.12-gcc11.2.0
Tcl/Tk 8.6.6	Intel Parallel Studio XE 2017 Update 2	/fs00/software/tcl/8.6.6-ips2017u2	tcl/8.6.6-ips2017u2
Tcl/Tk 8.6.4		/fs00/software/tcl/8.6.4	tcl/8.6.4
Tcl/Tk 8.6.4	Intel Parallel Studio XE 2016 Update 2	/fs00/software/tcl/8.6.4-ips2016u2	tcl/8.6.4-ips2016u2
Tcl/Tk 8.6.4	Intel Parallel Studio XE 2016 Update 2	/fs00/software/tcl/8.6.4-ips2016u2-avx2	tcl/8.6.4-ips2016u2-avx2

□□□□

□□□□□□□□ **License**□□□□□□□□□□□

FFTW 3.3.7

/fs00/software/fftw/3.3.7-iccifort-17.0.6-* □□ iccifort/17.0.6

FFTW 3.3.8

/fs00/software/fftw/3.3.8-ips2019u5 □□ ips/2019u5

LAMMPS 11Aug17

/fs00/software/lammps/11Aug17

LAMMPS 23Jun2022

/fs00/software/lammps/23Jun2022

LAMMPS 22Jul2025

/fs00/software/lammps/22Jul2025

OpenFOAM® v1806

/fs00/software/openfoam/v1806-ips2017u6

source /fs00/software/openfoam/v1806-ips2017u6/OpenFOAM-v1806/etc/bashrc

P4vasp 0.3.29

/fs00/software/p4vasp/0.3.29

Modulefile: p4vasp/0.3.29

Phonopy 1.11.2

/fs00/software/phonopy/1.11.2

Quantum ESPRESSO 5.2.0 & 6.1

/fs00/software/qe/5.2.0-ips2015u3/ □□ ips/2015u3

/fs00/software/qe/6.1-ips2017u2/ □□ ips/2017u2

ShengBTE

/fs00/software/shengbte □□ iccifort/15.0.3 openmpi/1.10.0-iccifort-15.0.3

Siesta 3.2-pl-5

/fs00/software/siesta/3.2-pl-5 □□ ips/2017u6

thirdorder 1.0.2 04d3f46feb78

/fs00/software/thirdorder/1.0.2

Modulefile: thirdorder/1.0.2 □□ anaconda/2-4.3.1 spglib/1.9.9

TBPLaS

/fs00/software/tbplas

MKL+MPI Version

Modulefile:

- oneapi/2024.0/compiler/2024.0.2
- oneapi/2024.0/ifort/2024.0.2
- oneapi/2024.0/mkl/2024.0
- oneapi/2024.0/mpi/2021.11

VASP6 GPU 编译

VASP6 GPU（ N卡） 编译实例

王勇 （孙建课题组 ）

人工微结构科学与技术协同创新中心高性能计算中心

根据 vasp官方给出的信息，以后 vasp gpu版本会着重开发 openacc标准的版本，之前老 cuda 版本会逐渐被废弃，因此我们以 vasp6.2 的 openacc版本结合官方指导进行编译。 官方指导链接

```
https://www.vasp.at/wiki/index.php/OpenACC_GPU_port_of_VASP
```

编译器：

Openacc gpu Vasp6官方给出编译器意见为使用 NVIDIA HPC-SDK或者 PGI's Compilers & Tools (version >=19.10)。在此 vasp官方建议使用 NVIDIA HPC-SDK且版本号最好为 **20.9** ，因为之后的版本可能会有一些对于 vasp的 bug。

关于 NVIDIA HPC-SDK 20.9的安装，见

```
https://developer.nvidia.com/nvidia-hpc-sdk-209-downloads
```

相关主页，安装流程十分简单，使用 wget下载，一键式安装，对于不连接外网的节点，可以手动下载 tarball 压缩包，上传后本地解压，在此不再赘述，安装前请使用 nvidia-smi命令查看本地硬件驱动和兼容的 cuda 版本（ version >=10.0），确保匹配，必要时候需升级硬件驱动。

安装 HPC-SDK 20.9的过程中会询问安装路径，我们以路径为 `/usr/software/nv-hpcsdk` 为例，在安装过程中指定此路径为例以后，需要设定环境变量:

```
export NVARCH=`uname -s`_`uname -m`;
export NVCOMPILERS=/usr/software/nv-hpcsdk #修改此处为安装路径
export PATH=$NVCOMPILERS/$NVARCH/20.9/compilers/bin:$PATH
export MANPATH=$MANPATH:$NVCOMPILERS/$NVARCH/20.9/compilers/man
export LD_LIBRARY_PATH=$NVCOMPILERS/$NVARCH/20.9/compilers/lib/:$LD_LIBRARY_PATH
export PATH=$NVCOMPILERS/$NVARCH/20.9/comm_libs/mpi/bin/:$PATH
```

上述可以每次使用 gpu vasp6时手动添加在任务脚本，也可直接写在 bashrc，但是不建议直接写在 bashrc ，可能会和 intel版本的 mpirun冲突，造成其他之前软件的运行问题。

依赖库：

安装完 NVIDIA HPC-SDK 20.9以后，还需要将软件运行需要的依赖库整理好，主要有 CUDA Toolkit, QD, NCCL, 以及 FFTW, 前三项直接包含在 HPC-SDK，不需要单独安装。

对于 FFTW，最好不要用 nvhpc-sdk的编译器进行安装，如果设置了上面安装完 hpc-sdk的环境变量，请先用 GNU或者 intel
的编译器环境变量进行覆盖，不然可能会导致计算效率问题，可以自己编译，也可以直接使用集群内已经安装
好的版本，路径为 `/fs00/software/fftw/3.3.8-ips2019u5`

编译：

准备好编译器和依赖库以后，就可以进行编译了，进入 vasp6.2根目录，

```
cp arch/makefile.include.linux_nv_acc makefile.include`
```

可以使用

```
which nvfortran | awk -F /compilers/bin/nvfortran '{ print $$1 }'`
```

查看 nvfortran是否为 nv-hpc-sdk的路径，如有问题，可以重新添加一遍环境变量。

（注：最近 **vasp**官方又在官网添加了新的 **openacc+openmp**混编版本的 **makefile.include.linux_nv_acc+omp+mkl**来解决 **nccl**库限制 **openacc** 版本只能单进程运行的问题，通过 **openmp** 来提升单进程多线程的运算效率，但目前没有太多相关测试的数据来支撑 **openmp** 混编会提升很多并行速度，所以在此我仍以旧版本的 **makefile.include.linux_nv_acc** 为例，同时官方也在积极改进这个问题，后续应该也可以多进程运行）

makefile.include的内容需要修改几处，包括确认编译器的位置，明确依赖库的路径等。修改后的 makefile.include如下（需要注意和修改的位置后面有注释）：

```
#Precompiler options

CPP_OPTIONS= -DHOST=\"LinuxPGI\" \
          -DMPI -DMPI_BLOCK=8000 -DMPI_INPLACE -Duse_collective \
          -DscaLAPACK \
```

```
          -DCACHE_SIZE=4000 \
          -Davoidalloc \
          -Dvasp6 \
          -Duse_bse_te \
          -Dtbdyn \
          -Dqd_emulate \
          -Dfock_dblbuf \
          -D_OPENACC \
          -DUSENCCL -DUSENCCLP2P

CPP         = nvfortran -Mpreprocess -Mfree -Mextend -E $(CPP_OPTIONS) $*$(FUFFIX)  >
$*$(SUFFIX)

FC          = mpif90 -acc -gpu=cc60,cc70,cc80,cuda11.0
FCL         = mpif90 -acc -gpu=cc60,cc70,cc80,cuda11.0 -c++libs

FREE        = -Mfree

FFLAGS      = -Mbackslash -Mlarge_arrays

OFLAG       = -fast

DEBUG       = -Mfree -O0 -traceback

#Specify your NV HPC-SDK installation, try to set NVROOT automatically
NVROOT      =$(shell which nvfortran | awk -F /compilers/bin/nvfortran '{ print $$1 }')
#or set NVROOT manually
#NVHPC      ?= /opt/nvidia/hpc_sdk
#NVVERSION  = 20.9
#NVROOT     = $(NVHPC)/Linux_x86_64/$(NVVERSION)

#Use NV HPC-SDK provided BLAS and LAPACK libraries
BLAS        = -lblas
LAPACK      = -llapack

BLACS       =
SCALAPACK   = -Mscalapack

CUDA        = -cudalib=cublas,cusolver,cufft,nccl -cuda
```

```
LLIBS      = $(SCALAPACK) $(LAPACK) $(BLAS) $(CUDA)

#Software emulation of quadruple precsion
QD         = $(NVROOT)/compilers/extras/qd                    #注意并修改这里
LLIBS      += -L$(QD)/lib -lqdmod -lqd
INCS       += -I$(QD)/include/qd

#Use the FFTs from fftw
FFTW       = /fs00/software/fftw/3.3.8-ips2019u5              #修改fftw路径至本地安装路径
LLIBS      += -L$(FFTW)/lib -lfftw3
INCS       += -I$(FFTW)/include

OBJECTS    = fftmpiw.o fftmpi_map.o fftw3d.o fft3dlib.o

#Redefine the standard list of O1 and O2 objects
SOURCE_O1  := pade_fit.o
SOURCE_O2  := pead.o

#For what used to be vasp.5.lib
CPP_LIB    = $(CPP)
FC_LIB     = nvfortran
CC_LIB     = nvc
CFLAGS_LIB = -O
FFLAGS_LIB = -O1 -Mfixed
FREE_LIB   = $(FREE)

OBJECTS_LIB= linpack_double.o getshmem.o

#For the parser library
CXX_PARS   = nvc++ --no_warnings

#Normally no need to change this
SRCDIR     = ../../src
BINDIR     = ../../bin
```

此外编译前和每次提交任务前请清除其他环境并重新设置一遍环境变量:

```
module load ips/2019u5  #这是为了我们使用集群安装好的fftw找到依赖的路径
export NVARCH=`uname -s`_`uname -m`;
export NVCOMPILERS=/usr/software/nv-hpcsdk
```

```
export PATH=$NVCOMPILERS/$NVARCH/20.9/compilers/bin:$PATH
export MANPATH=$MANPATH:$NVCOMPILERS/$NVARCH/20.9/compilers/man
export LD_LIBRARY_PATH=$NVCOMPILERS/$NVARCH/20.9/compilers/lib/:$LD_LIBRARY_PATH
export PATH=$NVCOMPILERS/$NVARCH/20.9/comm_libs/mpi/bin/:$PATH
```

`/usr/software/nv-hpcsdk` nv-hpc-sdk

```
make std gam ncl
```

openacc make gpu vasp_std GPU

1. **nccl** **openacc** **gpu** **.**

2.INCAR **NCORE** **openacc** **1.**

3.INCAR **NSIM** **KPAR** **KPAR** **GPU** **NSIM** **cpu** **.**

GPU

VASP

Quantum Espresso

□□ Apptainer

□□□□ qe7.1□□□□□□□□□□□□□□ `FeO_test.in`□□□ 1□ GPU□□□ `723090ib`□□□□□ `pw.x`

```
#BSUB -J FeO_test
#BSUB -q 723090ib
#BSUB -gpu num=1

export OMP_NUM_THREADS="$LSB_DJOB_NUMPROC"
QE="apptainer run --nv /fs00/software/singularity-images/ngc_quantum_espresso_qe-7.1.sif"
${QE} pw.x < FeO_test.in > FeO_test.out
```

NAMD

NAMD 2.12 (2016-12-22)

- /fs00/software/namd/2.12

□□□□□□□□□□□□□ in.conf□□□ 48□□□□□ e5v3ib□□

```
#BSUB -n 48
#BSUB -q e5v3ib

input=in.conf
#bindir=/fs00/software/namd/2.12/verbs/
bindir=/fs00/software/namd/2.12/ibverbs/

nodefile=nodelist
echo "group main" > $nodefile
for i in `echo $LSB_HOSTS`
do
□echo "host $i" >> $nodefile
done

${bindir}charmrun ++remote-shell ssh ++nodelist $nodefile +p$LSB_DJOB_NUMPROC ${bindir}namd2
$input
```

OOMMF

The Object Oriented MicroMagnetic Framework (OOMMF) 1.2 alpha 6

- /fs00/software/oommf/12a6-tcl8.6.4-ips2016u2 □□ tcl/8.6.4-ips2016u2□ ips/2016u2
- /fs00/software/oommf/12a6-tcl8.6.4-ips2016u2-avx2 □□ tcl/8.6.4-ips2016u2-avx2□ ips/2016u2

□□□□□□□□□□□□ sample.mif□□□ 72□□□□□ 6140ib□□

```
#BSUB -q 6140ib
#BSUB -n 72

module load ips/2016u2
module load tcl/8.6.4-ips2016u2-avx2

oommfin=sample.mif

oommfrun=/fs00/software/oommf/12a6-tcl8.6.4-ips2016u2-avx2/oommf.tcl

OOMMF_HOSTPORT=`tclsh $oommfrun launchhost 0`
export OOMMF_HOSTPORT
tclsh $oommfrun mmArchive
tclsh $oommfrun boxsi -numanodes auto -threads $LSB_DJOB_NUMPROC $oommfin
tclsh $oommfrun killoommf all
```

Mumax3

mumax 3

mumax3 版本	安装路径	建议加载的 **CUDA** 模块	可执行文件最高 **GLIBC** 符号	运行环境要求
3.10-cuda11.0	`/fs00/software/mumax/3.10-cuda11.0`	`cuda/11.2.0`	`GLIBC_2.4`	兼容老系统（ CentOS 7 ）
3.11.1-cuda12.3	`/fs00/software/mumax/3.11.1-cuda12.3`	`cuda/12.3.1`	`GLIBC_2.3.2`	在 **Rocky 9** 节点编译，建议在 Rocky 9 节点运行
3.11.1-cuda12.8	`/fs00/software/mumax/3.11.1-cuda12.8`	`cuda/12.8.1-el9`	`GLIBC_2.34`	必须 在 Rocky 9 / RHEL 9 等较新系统运行

均需要对应的 `module load cuda/<版本号>` 。

例: 输入文件为当前目录下的 sample.mx3, 申请 1个 GPU提交到 723090ib队列

```
#BSUB -q 723090ib
#BSUB -gpu num=1

mx3in=sample.mx3

module load cuda/11.2.0
/fs00/software/mumax/3.10-cuda11.0/mumax3 $mx3in
```

特别注意: CUDA 12 以上的版本在 Rocky 9节点编译，因此需要确认是否为 `login9` 节点。

此外对应软件目录下有对应的数据后处理软件如 `mumax3-convert` 和 `mumax3-plot` 。

□□□□

□□□□□□□□ **License**□□□□□□□□□□

AWS CLI v2

Name	Path	Module
AWS CLI current	/fs00/software/aws-cli/v2/current	aws-cli/current
AWS CLI 2.9.6	/fs00/software/aws-cli/v2/2.9.6	aws-cli/2.9.6

bbcp

Name	Path	Module
bbcp 14.04.14.00.1	/fs00/software/bbcp/14.04.14.00.1	bbcp/14.04.14.00.1

Boost

Name	Path	Module
Boost 1.72.0	/fs00/software/boost/1.72.0	boost/1.72.0
Boost 1.58.0	/fs00/software/boost/1.58.0	boost/1.58.0

CMake

Name	Path	Module
CMake 3.23.2	/fs00/software/cmake/3.23.2/	cmake/3.23.2
CMake 3.16.3	/fs00/software/cmake/3.16.3/	cmake/3.16.3
CMake 3.11.4	/fs00/software/cmake/3.11.4/	cmake/3.11.4

Git

Name	Path	Module

Git 2.38.1	/fs00/software/git/2.38.1	git/2.38.1

Grace

Name	Path	Module
Grace 5.1.25	/fs00/software/grace/5.1.25	grace/5.1.25

HDF5

Name	Path	Module
HDF5 1.10.5	/fs00/software/hdf5/1.10.5	hdf5/1.10.5

libpng

Name	Path	Module
libpng 1.5.26	/fs00/software/libpng/1.5.26	libpng/1.5.26

jq

Name	Path	Module
jq 1.7	/fs00/software/jq/1.7	jq/1.7

Libxc

Name	Compiler	Path	Module
Libxc 5.2.2	GNU Compiler Collection (GCC) 11.2.0	/fs00/software/libxc/5.2.2	libxc/5.2.2

libzip

Name	Path	Module
libzip 1.6.1	/fs00/software/libzip/1.6.1	libzip/1.6.1

NetCDF-C

Name	Path	Module
NetCDF-C 4.7.0	/fs00/software/netcdf/c-4.7.0	netcdf/c-4.7.0

PCRE

Name	Path	Module
PCRE 8.39	/fs00/software/pcre/8.39	pcre/8.39

Qt

Name	Path	Module
Qt 5.11.1	/fs00/software/qt/5.11.1	qt/5.11.1

rclone

Name	Path	Module
rclone latest	/fs00/software/rclone/latest	rclone/latest
rclone 1.66.0	/fs00/software/rclone/1.66.0	rclone/1.66.0

Spglib (OpenMP)

Name	Compiler	Path	Module
Spglib 1.9.9		/fs00/software/spglib/1.9.9	spglib/1.9.9
Spglib 1.9.0	GNU Compiler Collection (GCC) 5.2.0	/fs00/software/spglib/1.9.0-gcc5.2.0	spglib/1.9.0-gcc5.2.0

tmux

Name	Path	Module
tmux 3.3a	/fs00/software/tmux/3.3a	tmux/3.3a

zlib

Name	Path	Module
zlib 1.2.11	/fs00/software/zlib/1.2.11	zlib/1.2.11

Singularity Image

/fs00/software/singularity-images/

□□□□□□□□□□

gnuplot

gnuplot

Name	Path	Module	□□
gnuplot 6.0.5-el9	/fs00/software/gnuplot/6.0.5-el9	gnuplot/6.0.5-el9	□ qt □□□□□□□□□□□□□□□□
gnuplot 6.0.4-el9-gui	/fs00/software/gnuplot/6.0.4-el9	gnuplot/6.0.4-el9-gui	□ qt5 □□□□□□□□□□□□□□□□
gnuplot 5.2.7	/fs00/software/gnuplot/5.2.7	gnuplot/5.2.7	□ qt4 □□□□□□□□□□□□□□□□
gnuplot 5.2.2	/fs00/software/gnuplot/5.2.2	gnuplot/5.2.2	□ qt □□□□□□□□□□□□□□□□
gnuplot 5.0.6	/fs00/software/gnuplot/5.0.6	gnuplot/5.0.6	□ qt □□□□□□□□□□□□□□□□
gnuplot 5.0.1	/fs00/software/gnuplot/5.0.1	gnuplot/5.0.1	□ X11/qt □□□□□□□□□□□□□□□□

OVITO

OVITO

Name	Path	Module
OVITO 3.7.12	/fs00/software/ovito/3.7.12	ovito/3.7.12
OVITO 2.9.0	/fs00/software/ovito/2.9.0	ovito/2.9.0

Vim

Vim

Name	Path	Module
Vim 9.0.1677	/fs00/software/vim/9.0.1677	vim/9.0.1677
Vim 8.2.0488	/fs00/software/vim/8.2.0488	vim/8.2.0488
Vim 8.1	/fs00/software/vim/8.1	vim/8.1

Zsh

Name	Path	Module
Zsh latest	/fs00/software/zsh/latest	zsh/latest
Zsh 5.8	/fs00/software/zsh/5.8	zsh/5.8

或从 镜像站____下载编译安装

Environment Modules

添加 Environment Modules 至 Zsh

```
echo "source /fs00/software/modules/latest/init/profile.sh" >> ~/.zshrc
```

命令行提示符

- 因登录时需判断命令行提示符是否为 `$`，因此建议保持 Bash 为默认 Shell，登录后通过下列命令切换至 Zsh

```
module load zsh/latest && exec zsh
```

- 如设置 Zsh 为默认 Shell 或登录会自动切换至 Zsh，则必须更改 Zsh 的命令行提示符（环境变量 PS1）以 `$` 结尾（ $+空格），否则登录会失败!

Oh My Zsh

安装

在本地克隆后获取安装脚本再进行安装

```
git clone https://mirror.nju.edu.cn/git/ohmyzsh.git
cd ohmyzsh/tools
REMOTE=https://mirror.nju.edu.cn/git/ohmyzsh.git sh install.sh
```

切换已有 Oh My Zsh 至镜像源

```
git -C $ZSH remote set-url origin https://mirror.nju.edu.cn/git/ohmyzsh.git
git -C $ZSH pull
```

升级

```
omz update
```

主题

Powerlevel10k

安装

```
git clone --depth=1 https://mirror.nju.edu.cn/git/powerlevel10k.git ${ZSH_CUSTOM:-$HOME/.oh-my-zsh/custom}/themes/powerlevel10k
```

编辑 `~/.zshrc` 的主题 `ZSH_THEME="powerlevel10k/powerlevel10k"`

升级

```
cd ${ZSH_CUSTOM:-$HOME/.oh-my-zsh/custom}/themes/powerlevel10k && git pull
```

256色终端

开启 xterm 终端的 256色以便正确显示提示颜色

```
echo "export TERM=xterm-256color" >> ~/.zshrc
```

永久在左侧显示主机名

在集群中经常需要登录到各个节点，因此最好在左侧永久显示主机名防止误操作 编辑 `~/.p10k.zsh`

1. 将 `POWERLEVEL9K_RIGHT_PROMPT_ELEMENTS` 中的 `context` 剪切到 `POWERLEVEL9K_LEFT_PROMPT_ELEMENTS` 的第一项
2. 注释 `typeset -g POWERLEVEL9K_CONTEXT_{DEFAULT,SUDO}_{CONTENT,VISUAL_IDENTIFIER}_EXPANSION=` 这一行
3. 更改 `typeset -g POWERLEVEL9K_CONTEXT_PREFIX=` 为 `''`

插件

zsh-autosuggestions

□□

```
git clone https://mirror.nju.edu.cn/git/zsh-autosuggestions ${ZSH_CUSTOM:-~/.oh-my-zsh/custom}/plugins/zsh-autosuggestions
```

□□ `~/.zshrc` □ `plugins=` □□□ `zsh-syntax-highlighting`

```
plugins=( ... zsh-autosuggestions)
```

□□

```
cd ${ZSH_CUSTOM:-~/.oh-my-zsh/custom}/plugins/zsh-autosuggestions && git pull
```

zsh-syntax-highlighting

□□

```
git clone https://mirror.nju.edu.cn/git/zsh-syntax-highlighting.git ${ZSH_CUSTOM:-~/.oh-my-zsh/custom}/plugins/zsh-syntax-highlighting
```

□□ `~/.zshrc` □ `plugins=` □□□ `zsh-syntax-highlighting`

```
plugins=( [plugins...] zsh-syntax-highlighting)
```

□□

```
cd ${ZSH_CUSTOM:-~/.oh-my-zsh/custom}/plugins/zsh-syntax-highlighting && git pull
```

AlphaFold 2

`/fs00/software/alphafold/2.3.2/AlphaFold-v2.3.2.sif`

```
#BSUB -J alphafold
#BSUB -q gpu
#BSUB -n 16
#BSUB -gpu num=1

export OMP_NUM_THREADS="$LSB_DJOB_NUMPROC"
export MKL_NUM_THREADS="$LSB_DJOB_NUMPROC"

ALPHAFOLD_DATADIR=/fsb/data/alphafold/2
ALPHAFOLD_IMAGE=AlphaFold-v2.3.2.sif

ALPHAFOLD_APP=/app/alphafold

RUN_ALPHAFOLD="apptainer run --bind ${ALPHAFOLD_DATADIR}:${ALPHAFOLD_APP}/download \
                             --nv ${ALPHAFOLD_IMAGE}"

${RUN_ALPHAFOLD} python ${ALPHAFOLD_APP}/run_alphafold.py --use_gpu_relax \
    --data_dir=${ALPHAFOLD_APP}/download \
    --uniref90_database_path=${ALPHAFOLD_APP}/download/uniref90/uniref90.fasta \
    --mgnify_database_path=${ALPHAFOLD_APP}/download/mgnify/mgy_clusters_2022_05.fa \
    --
bfd_database_path=${ALPHAFOLD_APP}/download/bfd/bfd_metaclust_clu_complete_id30_c90_final_seq.
sorted_opt \
    --pdb70_database_path=${ALPHAFOLD_APP}/download/pdb70/pdb70 \
    --uniref30_database_path=${ALPHAFOLD_APP}/download/uniref30/UniRef30_2021_03 \
    --template_mmcif_dir=${ALPHAFOLD_APP}/download/pdb_mmcif/mmcif_files \
    --obsolete_pdbs_path=${ALPHAFOLD_APP}/download/pdb_mmcif/obsolete.dat \
    --model_preset=monomer \
    --max_template_date=2022-10-1 \
    --db_preset=full_dbs \
    --output_dir=output \
    --fasta_paths=input/input.fasta
```

AlphaFold 3

`/fs00/software/alphafold/3.0.0/AlphaFold-v3.0.0.sif`

0. 基本信息

alphafold3是 Google DeepMind开发的、用于生物分子复合体结构预测的人工智能模型，该模型接受一个 json 形式的、用于描述生物分子体系的文件，输出预测结构与置信度打分信息。结构预测过程主要分为两步：

- data pipeline:
  在一定参数控制下，基于输入的序列信息进行生物信息学搜索，得到结构预测所需的共进化信息 (multi-sequence alginment, msa) 与结构模板 (templates)，并将这些信息与输入信息整合，将整合信息作为 json文件输出
- inference: 读取整合入 msa和 templates的 json文件，转化为模型输入特征进行推理，输出 mmcif 格式的预测结构与 confidence score

此处部署的是基于 官方 github库打包的 apptainer容器。容器内包含 alphafold3 程序本体和依赖，不包含程序运行所需的数据库与预训练权重 ——

这些需要在运行时以命令行参数传入。集群上部署有生信搜索所需数据库，模型权重的使用受 term of use 控制，请自行向 Google DeepMind申请。

1. 使用方法

为避免冗长的路径和指令重复出现，定义变量

```
path_to_af3db="/fsb/data/alphafold/3"
path_to_af3container="/fs00/software/alphafold/3.0.0/AlphaFold-v3.0.0.sif "

io_dir=</path/to/your/input/and/output>
weights_dir=<path/to/directory/containing/your/af3-weights>

RUN_ALPHAFOLD="apptainer run --nv --bind
${path_to_af3db}:/databases,${io_dir}:/host_iopath,${weights_dir}:/af3_weights
${path_to_af3container} python run_alphafold.py"
```

其中 `${path_to_af3db}` 是集群上 alphafold3数据库地址，`${io_dir}` 是输入文件所在的、及希望结果输出的目录，`${weights_dir}` 是自己申请的 alphafold3权重文件所在目录。

`run_alphafold.py` 是 官方 github库 提供的接口文件。本页面后续出现的指令均默认这些变量存在。

输出帮助信息 (包括参数说明)

```
${RUN_ALPHAFOLD} --help
```

将输入文件 `input.json` 放在 `${io_dir}` 目录下，如下指令可使用 alphafold3 预测此文件描述体系的结构

```
${RUN_ALPHAFOLD} --db_dir=/databases/ --model_dir=/af3_weights/ \
  --json_path=/host_iopath/input.json --output_dir=/host_iopath/
```

程序将根据 `input.json` 中的 name 字段值，在 `${io_dir}` 下创建同名子目录，data pipeline 结束时补全的 json 文件、预测结构文件和 confidence score 都会被保存在这个子目录中。运行时 log 信息通过 stderr 输出（这是 python logging 模块的行为）。关于更详细的输入输出文件说明，请参考官方 input documentation 和 output documentation。

2. 分步运行 data pipeline 与 inference

如前所述，程序运行主要分为 data pipeline 和 inference 两步。其中，data pipeline 只使用 cpu 计算且耗时很长，如果在 gpu 队列上运行会导致 gpu 资源长期空置，可能导致高额费用和 gpu 资源浪费。因此，推荐先在 **cpu** 队列上运行 **data pipeline**，得到中间结果（也就是补全的 **json**）退出，然后再到 **gpu** 队列上进行结构预测。

为满足上述需求，可以用 bool 型命令行参数 `--run_data_pipeline` 和 `--run_inference` 控制模型只运行其中一步，两参数默认值均为 True.

分步运行指令示例：

- data pipeline (cpu job)

  ```
  ${RUN_ALPHAFOLD} --db_dir=/databases/ --model_dir=/af3_weights/ \
    --json_path=/host_iopath/input.json --output_dir=/host_iopath/ \
    --run_inference=False
  ```

- inference (gpu job)

  ```
  ${RUN_ALPHAFOLD} --db_dir=/databases/ --model_dir=/af3_weights/ \
    --json_path=/host_iopath/input.json --output_dir=/host_iopath/ \
    --run_data_pipeline=False
  ```

另外，出于研究或加速计算的目的，用户可能希望使用自行准备的 msa 或 templates，因而需要跳过 alphafold3 搜索步骤，此时需要手动将 msa 和 templates 写进输入 json 文件，然后带 `--run_data_pipeline=False` 参数运行模型，进行结构预测。关于如何将 msa 和 templates 写入 json 文件，请自行参考 DeepMind-input documentation。注意，msa 和 templates 会以文本形式写入 json 中，因此写入了 msa 和 templates 的 json 文件大小会达到数十到百 Mb 量级。

3. 关于其他命令行参数

除去以上参数外，容器还可接受很多其他参数，可以用 ${RUN_ALPHAFOLD} --help查看，或是在 run_alphafold.py中查看。我认为有 3点值得注意：

- - input_dir和 --json_path都可以指定输入文件，使用时只应指定其一。当指定 input_dir
时，模型会从 input_dir中寻找一个 json格式文件，当作输入文件
- - jackhmmer_n_cpu和 --nhmmer_n_cpu指定生信搜索中使用 cpu cores数目的上限
机器学习程序通常允许设置 random seed来保证运行的可重复性， alphafold3的random seed
不能以命令行参数指定，而是在 input.json中

4. 队列兼容性、测试中的运行耗时

data pipeline对设备要求不高， inference则可能对 gpu设备有些要求。目前测试了两例：

2PV7，蛋白，双链 homomer，单链长度 298
1AKE，蛋白，双链 homomer，单链长度 214

在队列 83a100ib和734090d上测试结果显示：以上体系在单张 A100卡 (memory 40 G) 或单张 4090d (memory 24 G) 上均可完成 inference. 在队列 722080tiib和72rtxib上无法成功运行 . 其他 GPU

队列未做测试。注意，官方 performance documentation文档指出官方仅测试过 alphafold3在A100(80G), A100(40 G)和 H100设备上推理的准确性，因此使用如 4090等其他设备做推理时，请自行校验结果准确性。

测试中 data pipeline和inference计算所需时间如下：

使用 8 cpu cores对一条长约 300的蛋白序列进行 msa和templates搜索耗时为 1.5-2 h；其中 ，搜出
msa后，根据部分匹配片段信息挑选出 templates耗时为 10 s量级，相对于总时间可以忽略
对每个随机种子， alphafold3会预测 5个结构并按置信度打分挑选最好的作为最终结果，此过程耗时
100 s左右； inference步总耗时与设置随机种子数有关

更多 performance信息可参考官方 performance documentation。

5. 运行结果准确性评估

比较 2PV7预测结构与实验结构，全原子 RMSD=4.410 (单位: Angstrom，下同)。
为衡量单链水平结构误差，计算预测 A:B链-实验 A:B链间 RMSD，4个 RMSD值分别为 3.012, 2.759, 2.971, 2.740. 此单链上， alphafold3预测准确度低于 alphafold2. 为衡量 2个subunits间相对位置误差，计算 DockQ 分数为 0.499>0.23，预测结构 docking成功。

1AKE双链结构预测有较大错误，预测结构与 1AKE实验结构 RMSD=18.176，与 4AKE实验结构 RMSD=26.791 ，该结果远不如 alphafold2-multimer。subunits间DockQ=0.019<<0.23
，说明结构相对位置也预测错误。尚不清楚是 Alphafold3本身原因，还是部署或使用不当导致。使用 4090d和 A100推理的结果无明显差异。

dockQ (Mirabello & Wallner, 2024, Bioinformatics) 打分用于衡量模型对 subunits “ 间位置关系预测准确度。此分数取值范围是 0-1，通常 <0.23认为错误， >0.8认为准确。

RoseTTAFold

`/fs00/software/rosettafold/1.1.0`

□□□ `apptainer`□□□ bind□□□ `home`□□□□□□□□□□□ `RoseTTAFold`□□ `folding`□ conda □□□□□□□□□□□□□□□□□□□ `--no-home`□□□□□□□□

```
#BSUB -J RoseTTAFold
#BSUB -q gpu
#BSUB -n 8
#BSUB -gpu num=1

###### Configurie Numpy threads  ######

export OMP_NUM_THREADS="$LSB_DJOB_NUMPROC"
export MKL_NUM_THREADS="$LSB_DJOB_NUMPROC"

###### Definition ######

ROSETTAFOLD_DATADIR=/fsb/data/rosettafold      # path to RoseTTAFold data (host)
ROSETTAFOLD_IMAGE=RoseTTAFold-1.1.0.sif         # path to RoseTTAFold image (host)
ROSETTAFOLD_APPDIR=/app/RoseTTAFold             # path to RoseTTAFold working directory
(container)

###### Database ######

UNIREF30_DB=$ROSETTAFOLD_DATADIR/UniRef30_2020_06
BFD_DB=$ROSETTAFOLD_DATADIR/bfd
PDB100_DB=$ROSETTAFOLD_DATADIR/pdb100_2021Mar03

###### Example ######

RUN_ROSETTAFOLD="apptainer run --bind $UNIREF30_DB:$ROSETTAFOLD_APPDIR/UniRef30_2020_06 \
                --bind $BFD_DB:$ROSETTAFOLD_APPDIR/bfd \
                --bind $PDB100_DB:$ROSETTAFOLD_APPDIR/pdb100_2021Mar03 \
                --nv $ROSETTAFOLD_IMAGE"

# For monomer structure prediction (e2e)
```

```
${RUN_ROSETTAFOLD} $ROSETTAFOLD_APPDIR/run_e2e_ver.sh $ROSETTAFOLD_APPDIR/example/input.fa
output/

# For monomer structure prediction (pyrosetta)
${RUN_ROSETTAFOLD} $ROSETTAFOLD_APPDIR/run_pyrosetta_ver.sh
$ROSETTAFOLD_APPDIR/example/input.fa output/

# For complex modeling
${RUN_ROSETTAFOLD} python $ROSETTAFOLD_APPDIR/network/predict_complex.py \
                -i $ROSETTAFOLD_APPDIR/example/complex_modeling/paired.a3m \
                -o output/ -Ls 218 310

# For PPI screening using faster 2-track version (example input and output are at
example/complex_2track)
${RUN_ROSETTAFOLD} python $ROSETTAFOLD_APPDIR/network_2track/predict_msa.py \
                -msa $ROSETTAFOLD_APPDIR/example/complex_2track/input.a3m \
                -npz output/complex.npz -L1 218
```

ParallelFold 2

`/fs00/software/parallelfold/2.0-2.3.1/ParallelFold-v2.0-with-AlphaFold-v2.3.1.sif`

```
#BSUB -J parafold
#BSUB -q gpu
#BSUB -n 16
#BSUB -gpu num=1

export OMP_NUM_THREADS="$LSB_DJOB_NUMPROC"
export MKL_NUM_THREADS="$LSB_DJOB_NUMPROC"

ALPHAFOLD_DATADIR=/fsb/data/alphafold/2
PARAFOLD_IMAGE=ParallelFold-v2.0-with-AlphaFold-v2.3.1.sif

PARAFOLD_APP=/app/ParallelFold

RUN_PARAFOLD="apptainer run --bind ${ALPHAFOLD_DATADIR}:${PARAFOLD_APP}/download \
                            --nv ${PARAFOLD_IMAGE}"

${RUN_PARAFOLD} python ${PARAFOLD_APP}/run_alphafold.py --use_gpu_relax \
    --parameter_path=${PARAFOLD_APP}/download/params \
    --uniref90_database_path=${PARAFOLD_APP}/download/uniref90/uniref90.fasta \
    --mgnify_database_path=${PARAFOLD_APP}/download/mgnify/mgy_clusters_2022_05.fa \
    --
bfd_database_path=${PARAFOLD_APP}/download/bfd/bfd_metaclust_clu_complete_id30_c90_final_seq.s
orted_opt \
    --pdb70_database_path=${PARAFOLD_APP}/download/pdb70/pdb70 \
    --uniref30_database_path=${PARAFOLD_APP}/download/uniref30/UniRef30_2021_03 \
    --template_mmcif_dir=${PARAFOLD_APP}/download/pdb_mmcif/mmcif_files \
    --obsolete_pdbs_path=${PARAFOLD_APP}/download/pdb_mmcif/obsolete.dat \
    --model_preset=monomer \
    --max_template_date=2022-10-1 \
    --db_preset=full_dbs \
    --output_dir=output \
    --fasta_paths=${PARAFOLD_APP}/input/input.fasta
```