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         = $(NVROOT)/compilers/extras/qd                    #□□□□□□□□
LLIBS      += -L$(QD)/lib -lqdmod -lqd
INCS       += -I$(QD)/include/qd

#Use the FFTs from fftw
FFTW       = /fs00/software/fftw/3.3.8-ips2019u5              #□□fftw□□□□□□□□□□
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`为 nv-hpc-sdk的安装路径　　确认好以上所有设置后，就可以使用

```
make std gam ncl
```

进行编译了，注意，由于　　openacc版本的原因，直接取消了编译　　make gpu的方式，编译得到的　　vasp_std

之类的版本直接可以使用　　GPU进行计算。

其他注意事项:

1.由于　**nccl**库的限制，　　**openacc**的　**gpu**版本只能单进程运行　　.

2.INCAR NCORE openacc 1.

3.INCAR NSIM KPAR KPAR

GPU NSIM cpu .

GPU

VASP

Revision #8
Created 2021-11-16 16:15:14 CST by Yao Ge
Updated 2023-02-08 23:08:00 CST by Yao Ge