南京大学 AI 平台 GPU 节点

用户手册（试用版）

目录

一、环境介绍：

1、硬件环境：

集群配置了 5 台浪潮 NF5468 GPU 节点

-2 颗 Intel Xeon Glod 6248 CPU

-24*32GB ECC DDR4 2666MHz 内存

-4*2TB 企业级 SSD 硬盘（Raid6）

-8*GPU_NV_32GB_V100 卡

-4*100 HCA 卡

2、操作系统：

Red Hat Enterprise Linux Server release 7.7 64 位操作系统

3、作业管理系统：

Platform LSF 10.1

4、编译软件：

Intel C version 2018.5.274

Intel Fortran version 2018.5.274

gcc version 9.2.0

gcc version 7.4.0

openmpi version 2.1.6(intel 编译器编译)

openmpi version 3.1.4(intel 编译器编译)

5、常用软件：

常用软件安装目录：/share/apps/

1	anaconda3-4.4.0	21	MaterialsStudio8.0
2	anaconda3-5.3.1	22	jasper-2.0.6
3	cmake-3.16.0	23	jasper-2.0.14
4	fftw-3.3.6	24	mpich-3.1.2
5	fftw-3.3.8	25	mpich-3.3.1
6	gcc5.4.0	26	ncl-6.4.0

7	gcc7.4.0	27	ncl-6.5.0
8	gcc9.2.0	28	ncl-6.6.2
9	hdf5-1.10.4	29	netcdf-4.4
10	hdf5-1.10.4_intel2018	30	netcdf-4.4_intel
11	hdf5-1.10.4_intel2019	31	openmpi-2.1.6_intel
12	hdf5-1.8.20	32	openmpi-3.1.4_intel
13	intel_2013u2	33	python2.7.6
14	intel_2017u4	34	python3.8.0
15	intel_2018u1	35	perl-5.28.2
16	intel2018u2	36	perl-5.30.1
17	intel_2018u4	37	udunits-2.2.26
18	intel_2019u5	38	udunits-2.2.26_intel
19	matlab2015a	39	vasp-5.3.5
20	matlab2019b	40	zlib-1.2.11

6、环境变量：

```
module av
```

```
-------------------------------------------------------------

/share/apps/modulefiles

-------------------------------------------------------------

cmake/3.16.0          cudnn/10.0_v7.6.3     gcc/gcc-9.2.0
jpeg-9b               netcdf/4.4            udunits/2.2.26_intel
cuda/10.0             cudnn/10.1_v7.6.4     intel/2013u2
mpich/3.1.2           netcdf/4.4_intel      vasp-5.3.5
cuda/10.1             cudnn/8.0_v7.1        intel/2017u4
mpich/3.3.1           openmpi/2.1.6_intel
cuda/10.1.168         cudnn/9.0_v7.6.4      intel/2018u1
ncl/6.4.0             openmpi/3.1.4_intel
cuda/8.0              cudnn/9.2_v7.6.3      intel/2018u2
ncl/6.5.0             python/2.7.6
```

cuda/9.0	fftw/3.3.8	intel/2018u4
ncl/6.6.2	python/3.8.0	
cuda/9.2	gcc/gcc-7.4.0	intel/2019u5
ncview	udunits/2.2.26	

二、提交命令：

1、概述：

bsub -gpu - | "[num=*num_gpus*] [:mode=shared | exclusive_process] [:mps=yes | no | per_socket | per_gpu] [:j_exclusive=yes | no] [:aff=yes | no] [:gmodel=*model_name*[-*mem_size*]] [:gtile=*tile_num*|'!'] [:gmem=*mem_value*] [:nvlink=yes]"

2、常用参数介绍：

```
#bsub -gpu
```

```
num=num_gpus
```

#作业所需的物理 GPU 数量。 默认情况下，该数量是每个主机的数量。

```
mode=shared | exclusive_process
```

#作业运行时的 GPU 模式，即 shared 或 Exclusive_process。默认模式为共享。

```
nvlink=yes
```

#为 GPU 之间的 NVLink 连接启用作业执行。LSF 为 GPU 分配有效的 NVLink 连接。
#默认情况下，当没有足够的具有 NVLink 连接的 GPU 时，LSF 可以分配没有 NVLink 连接的 GPU。

```
aff=yes | no
```

#指定是否强制执行严格的 GPU-CPU 相似性绑定。如果设置为 no，则 LSF 放宽 GPU 亲和力，同时保持 CPU 亲和力。默认情况下，aff=yes 设置为维持严格的 GPU-CPU 关联性绑定。

```
gmodel =  short_model_name
```

#请求具有特定品牌名称（例如，Tesla，Quadro，NVS）或型号类型名称（例如，K40，v100）的 GPU。

3、提交示例：

```
bsub -q gpu_v100 -gpu “num=2” ./app
```

#提交一个需要 2 个物理 GPU 数量的作业至 gpu_v100 队列

```
bsub -q gpu_v100 -gpu “num=2:mode=share” ./app
```

#提交一个需要 2 个共享模式物理 GPU 数量的作业至 gpu_v100 队列(默认为共享模式)

```
bsub -q gpu_v100 -gpu “num=2:mode=exclusive_process” ./app
```

#提交一个需要 2 个独占模式物理 GPU 数量的作业至 gpu_v100 队列

```
bsub -q gpu_v100 -gpu “num=2:aff=yes” ./app
```

#提交一个需要 2 个物理 GPU 数量的作业至 gpu_v100 队列,进行 GPU-CPU 相似性绑定。

4、提交模板：

```
vim test.sh #编写提交脚本
#!/bin/bash
#BSUB -q gpu_v100#指定使用 gpu_v100 队列
#BSUB -J test#定义作业名
#BSUB -gpu "num=4"#定义使用几块 GPU
#BSUB -n 2#定义 Task 值
#BSUB -o %J.out #定义输出文件名
#BSUB -e %J.err#定义错误输出文件名

export I_MPI_HYDRA_BOOTSTRAP=lsf#对接 inetl 自动加载 lsf -n 值指定的 task 值
source
```

```
/share/apps/intel/intel_2017u4/impi/2017.3.196/bin64/mpivars.sh
source
/share/apps/intel/intel_2017u4/compilers_and_libraries_2017.4.196/lin
ux/bin/compilervars.sh intel64
source
/share/apps/intel/intel_2017u4/compilers_and_libraries_2017.4.196/lin
ux/mkl/bin/mklvars.sh intel64
export PATH=/share/gpu-apps/cuda/9.2/bin:$PATH
export
LD_LIBRARY_PATH=/share/gpu-apps/cuda/9.2/lib64:$LD_LIBRARY_PATH
#加载环境变量
mpiexec.hydra  ./vasp_gpu >> admin.out
bsub < test.sh#提交作业
```

5、查看 gpu 资源：

```
lshosts -gpu
```

#为 lshosts 选项显示集群 GPU 的拓扑信息

HOST_NAME	gpu_id	gpu_model	gpu_driver	gpu_factor	numa_id
gpu01	0	TeslaK40c	418.87.00	3.5	0
	1	TeslaK40c	418.87.00	3.5	1
gpu02	0	TeslaK40c	418.87.00	3.5	0
	1	TeslaK40c	418.87.00	3.5	1

```
lsload -gpu
```

#显示基于主机的 GPU 信息

HOST_NAME	status	ngpus	gpu_shared_avg_mut	gpu_shared_avg_ut	ngpus_physical
gpu01	ok	2	0%	0%	2
gpu02	ok	2	0%	0%	2

```
lsload -gpuload
```

#显示基于 GPU 的 GPU 信息

```
HOST_NAME          gpuid    gpu_model    gpu_mode   gpu_temp    gpu_ecc
gpu_ut    gpu_mut  gpu_mtotal  gpu_mused     gpu_pstate     gpu_status
gpu_error
gpu01                0    TeslaK40c          0.0         26C         0.0
0%          0%         11.1G            0M                 8                ok
-
                     1    TeslaK40c          0.0         27C         0.0
0%         0%        11.1G          0M              8             ok
```

三、应用软件：

集群已安装 tensorflow-gpu 1.8.0 版本，使用的环境变量是

```
export PATH=/share/apps/anaconda/3-4.4.0/bin:$PATH
module load cuda/9.0
module load cudnn/9.0_v7.6.4
```

使用命令：python yourfile.py

这里介绍下在集群上编译 GPU 版本 tensorflow 和 GPU 版本 vasp 的方法：

1、GPU 版本 tensorflow：

tensorflow 可以从 python 的模块里直接调用，因此需要提前安装 python

已测试，tensorflow-gpu 1.13.0 在集群上无法使用，依赖高版本的 glibc，建议安装低版本。

（1）安装 python3 或者 anaconda3

```
./Anaconda3-4.4.0-Linux-x86_64.sh
>> accept  ##接受协议
>>/share/home/yourname/apps/anaconda3  ##指定安装目录
```

（2）加载 python 环境变量，从官网下载 tensorflow-gpu 软件版，如.whl 格式

```
export PATH=/share/home/yourname/apps/anaconda3/bin:$PATH
pip install  tensorflow_gpu-1.8.0-cp36-cp36m-manylinux1_x86_64.whl
```

此时若依赖包没有安装完成，会提示缺少的依赖包名，根据依赖包名下载安装即可。

（3）使用：

使用时直接加载 python 环境变量即可

```
export PATH=/share/home/yourname/apps/anaconda3/bin:$PATH
```

并根据加载需要使用的 cuda 版本，可通过 module avail 查看

2、GPU 版本 vasp：

此时以编译 vasp 5.4.4 为例

（1）编译步骤：

加载 intel 环境变量：

```
source
/share/apps/intel/intel_2018u4/impi/2018.4.274/bin64/mpivars.sh
source
/share/apps/intel/intel_2018u4/compilers_and_libraries_2018.5.274/lin
ux/bin/compilervars.sh intel64
source
/share/apps/intel/intel_2018u4/compilers_and_libraries_2018.5.274/lin
ux/mkl/bin/mklvars.sh intel6
# tar -xvf vasp-5.4.4.tar.gz
# cd vasp.5.4.4
# cp arch/makefile.include.linux_intel makefile.include
# vim makefile.include   ##具体内容见下方
# make gpu  ## 在 bin 目录下会生产 vasp_gpu 可执行文件
```

makefile.include 文件内容如下，只需要修改红色位置，指定 cuda 目录即可
CPP_OPTIONS= -DHOST=\"LinuxIFC\"\ -DMPI -DMPI_BLOCK=8000 \ -Duse_collective \

```
              -DscaLAPACK \
              -DCACHE_SIZE=4000 \
              -Davoidalloc \
              -Duse_bse_te \
              -Dtbdyn \
              -Duse_shmem
CPP         = fpp -f_com=no -free -w0  $*$(FUFFIX) $*$(SUFFIX)
$(CPP_OPTIONS)
FC          = mpiifort
FCL         = mpiifort -mkl=sequential -lstdc++
FREE        = -free -names lowercase
FFLAGS      = -assume byterecl -w
OFLAG       = -O2
OFLAG_IN    = $(OFLAG)
DEBUG       = -O0
MKL_PATH    = $(MKLROOT)/lib/intel64
BLAS        =
LAPACK      =
BLACS       = -lmkl_blacs_intelmpi_lp64
SCALAPACK   = $(MKL_PATH)/libmkl_scalapack_lp64.a $(BLACS)
OBJECTS     = fftmpiw.o fftmpi_map.o fft3dlib.o fftw3d.o
$(MKLROOT)/interfaces/fftw3xf/libfftw3xf_intel.a
INCS        =-I$(MKLROOT)/include/fftw
LLIBS       = $(SCALAPACK) $(LAPACK) $(BLAS)
OBJECTS_O1 += fftw3d.o fftmpi.o fftmpiw.o
OBJECTS_O2 += fft3dlib.o
CPP_LIB     = $(CPP)
FC_LIB      = $(FC)
CC_LIB      = icc
```

```
CFLAGS_LIB = -O
FFLAGS_LIB = -O1
FREE_LIB   = $(FREE)
OBJECTS_LIB= linpack_double.o getshmem.o
CXX_PARS   = icpc
LIBS       += parser
LLIBS      += -Lparser -lparser -lstdc++
SRCDIR     = ../../src
BINDIR     = ../../bin
CPP_GPU    = -DCUDA_GPU -DRPROMU_CPROJ_OVERLAP -DUSE_PINNED_MEMORY
-DCUFFT_MIN=28 -UscaLAPACK
OBJECTS_GPU = fftmpiw.o fftmpi_map.o fft3dlib.o fftw3d_gpu.o
fftmpiw_gpu.o
CC         = icc
CXX        = icpc
CFLAGS     = -fPIC -DADD_ -Wall -qopenmp -DMAGMA_WITH_MKL
-DMAGMA_SETAFFINITY -DGPUSHMEM=300 -DHAVE_CUBLAS
CUDA_ROOT  ?= /share/gpu-apps/cuda/10.0
NVCC       := $(CUDA_ROOT)/bin/nvcc -ccbin=icc
CUDA_LIB   := -L$(CUDA_ROOT)/lib64 -lnvToolsExt -lcudart -lcuda -lcufft
-lcublas
GENCODE_ARCH    := -gencode=arch=compute_30,code=\"sm_30,compute_30\"
\
                   -gencode=arch=compute_35,code=\"sm_35,compute_35\"
\
                   -gencode=arch=compute_60,code=\"sm_60,compute_60\"
MPI_INC    = $(I_MPI_ROOT)/include64/
```

（2）使用

加载 intel 环境变量和使用的 cuda 环境变量：

```
source
/share/apps/intel/intel_2018u4/impi/2018.4.274/bin64/mpivars.sh
source
/share/apps/intel/intel_2018u4/compilers_and_libraries_2018.5.274/lin
ux/bin/compilervars.sh intel64
source
/share/apps/intel/intel_2018u4/compilers_and_libraries_2018.5.274/lin
ux/mkl/bin/mklvars.sh intel6
module load cuda/10.0
mpirun -np 4 ./vasp_gpu
```

##vasp 在 GPU 上运行时，需要设置 INCAR 文件中 LREAL=.T.