1 软件介绍

LAMMPS 即 Large-scale Atomic/Molecular Massively Parallel Simulator,可以翻译为大规模原子分子并 行模拟器,主要用于分子动力学相关的一些计算和模拟工作,一般来讲,分子动力学所涉及到的领域, LAMMPS 代码也都涉及到了。

2 安装环境

lammps版本: lammps-16Mar18

操作系统: Ubuntu 16.04

编译器: GCC

cuda: 9.0

mpi: openmpi-3.0-gnu

3 安装步骤

3.1 CUDA

到 <u>https://developer.nvidia.com/cuda-downloads</u>下载对应操作系统的 cuda 安装包。下载后执行:

```
chmod +x cuda_9.0.176_384.81_linux.run #使之具有可执行权限
sudo sh cuda 9.0.176 384.81 linux.run
```

然后按照相关的提示输入安装路径即可,本文选择默认路径。详细安装步骤可以参考 CUDA 9 安

装手册。

环境变量:

```
cat /etc/profile.d/cuda-env.sh
export PATH=/usr/local/cuda-9.0/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-9.0/lib64:$LD_LIBRARY_PATH
export C INCLUDE PATH=/usr/local/cuda-9.0/include:$C INCLUDE PATH
```

3.2 MPI

目前 lammps 在 gpu 环境下选择 openmpi-3.0-gnu 可以安装成功。下载链接: <u>https://www.open-</u>mpi.org/software/ompi/v3.0/

3.2.1 OpenMPI 简介

OpenMPI 是一个免费的、开源的 MPI 实现, 兼容 MPI-1 和 MPI-2 标准。OpenMPI 由开源社区开发 维护, 并具有很高的性能。

OpenMPI目前最新版本为 openmpi-3.0.0, 官方网站: <u>http://www.open-mpi.org/</u>, 可从官网免费下载 Openmpi 源码安装包。

CUDA-aware MPI: 支持 CUDA-aware 意味着 MPI 库可以直接接收和发送 GPU 显存数据,以下版本及之后的版本都支持。

- * MVAPICH2 1.8/1.9b
- * OpenMPI 1.7
- * CRAY MPI (MPT 5.6.2)
- * IBM Platform MPI (8.3)
- * SGI MPI (1.08)

3.2.2 安装 OpenMPI

以 OpenMPI 3.0.0 为例:

```
$ tar zxvf openmpi-3.0.0.tar.gz
$ cd openmpi-3.0.0
$ ./configure --prefix=<u>/public/software/mpi/openmpi-3.0.0</u> --enable-mpirun-
prefix-by-default --without-psm CC=gcc CXX=g++ FC=gfortran F77=gfortran -
-with-cuda=/usr/local/cuda-9.0
$ make -j 8 #8个进程并行编译
```

\$ make install
设置环境变量脚本:

vim /public/software/profile.d/openmpi-3.0.0-env.sh

```
export MPI_HOME=/public/software/mpi/openmpi-3.0.0
export PATH=${MPI_HOME}/bin:${PATH}
export LD_LIBRARY_PATH=${MPI_HOME}/lib:${LD_LIBRARY_PATH}
export MANPATH=${MPI_HOME}/share/man:${MANPATH}
```

> Tips

1. OpenMPI 安装会自动检测编译节点本地可用的通信网络设备,如需支持 InfiniBand 网络,请确保编译 MPI 前该节点已安装 OFED 驱动。

2. 执行 OpenMPI 安装目录\$MPI_HOME/bin 下的 ompi_info 命令,可查询当前 OpenMPI 配置信息。

3.2.3 编译 MPI 程序

ī示:

语言类型	MPI 编译器
С	mpicc
C++	mpicxx
Fortran77	mpif77
Fortran90	mpif90

MPI 编译器是对底层编译器的一层包装,通过-show 参数可以查看实际使用的编译器。比如:

```
$ mpicc -show
```

```
icc -I/public/software/mpi/openmpi-3.0.0/include -pthread -
L/public/software/mpi/openmpi-3.0.0-intel/lib -lmpi -ldl -lm -lnuma -Wl,--
export-dynamic -lrt -lnsl -lutil
   编译程序示例:
```

\$ source /public/software/profile.d/openmpi-3.0.0-env.sh \$ mpicc -o hello hello.c \$ mpif90 -o hello hello.f90

3.2.4 运行 MPI 程序

3.2.4.1 基本命令

OpenMPI 使用自带的 OpenRTE 进程管理器, 启动命令为 mpirun/mpiexec/orterun, 基本格式如下:

```
$ mpirun -np N -hostfile <filename> <program>
```

- -np N: 运行 N 个进程 \geq
- -hostfile: 指定计算节点, 文件格式如下: \geq

```
node1 slots=8
```

```
node2 slots=8
```

slots=8代表可在该节点上执行8个进程,也可将 node1和 node2分别写8行。

3.2.4.2 选择通信网络

OpenMPI 支持多种通信协议,如以太网 TCP、IB 网络、内存共享等,可通过设置--mca btl 参数进 行选择。如:

\$ mpirun -np 12 -hostfile hosts --mca btl self,sm,openib ./program 将在节点内使用共享内存,节点间使用 IB 网络通信,这种方式通信效率较高。

▶ mca 参数说明

*	mca btl self,tcp	使用以太网 TCP/IP 协议通信
•	mca btl self,sm	单节点运行时使用共享内存,效率较高
*	mca btl self,openib	有 Infiniband 设备时,使用 IB 通信
*	mca btl_tcp_if_include eth0	以太网通信时使用 eth0 接口,默认同时使用所
有接口(包括 IPoIB)	
*	mca orte_rsh-agent rsh	指定节点间通信使用 rsh,默认为 ssh

> Tips

- ▲ --mca btl 参数中必须含有 self, 单进程自己与自己通信
- ▲ 可同时指定多个 btl 参数, 如--mca btl self, sm, openib, 最先出现者优先级较高

▲ IPoIB 设定: --mca btl self,tcp --mca btl_tcp_if_include ib0, 当然更直接的方式是

在 hostfile 里使用 IB 网卡的 IP 地址或对应的 hostname

3.3 Lammps 安装

下载 Lammps:

wget http://lammps.sandia.gov/tars/lammps-stable.tar.gz

解压下载文件:

tar -xzvf lammps-stable.tar.gz

cd lammps-16Mar18

编辑配置文件,在 Volta V100 环境下编译:

vi src/MAKE/OPTIONS/Makefile.kokkos_cuda_mpi # edit for volta

#KOKKOS_ARCH = Kepler35

 $KOKKOS_ARCH = Volta70$

编译 Lammps:

\$ cd src

\$ make clean-all

\$ make no-all

\$ make no-lib

\$ make yes-manybody yes-molecule yes-replica yes-kspace yes-asphere yes-rigid yes-misc yes-user-omp yes-user-reaxc

\$ make yes-kokkos

\$ make kokkos_cuda_mpi -j32

3.4 Lammps 基于 NGC Docker 安装

在 NVIDIA GPU Cloud (NGC) 中,包含 Lammps 的 docker 镜像文件,可以直接下载,导入 Linux docker 环境就可以使用。

注册 NGC

← → C a Secure https://ngc.nvidia.com/signin/email		☆ 🖓 🖾 🕼 :
	NIDIA. GPU CLOUD Sign In ©	
	Please enter your email address	
	Sign Up	

填写注册信息:

Sign Op		(
NAME	ROLE	
Please enter your full name	Please select your job role	N
COMPANY	INDUSTRY	
Please enter your company or organization	Please select your industry	N
EMAIL	COUNTRY	
Please enter your email address	Please select your country	×
 I agree to the <u>NVIDIA ACCOUNT TERMS C</u> By obtaining any third party software comv registration information with such t 	FUSE ontainers through NGC, I agree that NVIDI hird party software providers, who may us	A will shar se my

登录 NGC 系统

注册 NGC 后, 登录 NGC 系统, 就可以看到下图中所有的 HPC Apps 的 docker image 镜像。



Docker image 下载

下载 image 镜像之前, 先要获取 API key:

Registry		Get API Key
Documentation How to use NGC container	rs on supported platforms >	
Repositories	hpc/lammps	
\checkmark nvidia	docker pull nvcr.io/hpc/lammps:patch230ct2017	ſſ
∨ nvidia/k8s		

点击 Get API Key, 即可获得:

Configuration > API Key

API

API Information

Your API Key authenticates your use of NGC service when using NGC CLI or the Docker client. Anyone with this API Key has access to all services, actions, and resources on your behalf.

Click Generate API Key to create your own API Key. If you have forgotten or lost your API Key, you can come back to this page to create a new one at any time.

Usage

Use your API key to log in to the NGC registry as follows.

Docker[™] 🕝

For the username, enter '\$oauthtoken' exactly as shown. It is a special authentication token for all users.

\$ docker login nvcr.io			
Username: \$oauthtoken Password: <your key=""></your>			

点击"Generate API Key",并点击弹出对话框中的"Confirm",系统会生成一个 API Key 作为 nvcr.io 的登录密码,并复制该 Password,用于登录:

Configuration > API Key Generate API Key
API
API Information Your API Key authenticates your use of NGC service when using NGC CLI or the Docker client. Anyone with this API Key has access to all services, actions, and resources on your behalf.
Click Generate API Key to create your own API Key. If you have forgotten or lost your API Key, you can come back to this page to create a new one at any time.
Usage
Use your API key to log in to the NGC registry as follows.
Docker ^{ts} ြ
For the username, enter '\$oauthtoken' exactly as shown. It is a special authentication token for all users.
\$ docker login nvcr.io
Username: \$oauthtoken
Password: NDg0YzJ0ajIyMjRrdWxzcjNrzXB2dG40cWE6MTQ0ZTUxZTUtZGEwNC00ZTgwLWE1NjItoDgyNDkyoTJINzUz
API Key generated successfully. This is the only time your API Key will be displayed. Keep your API Key secret. Do not share it or store it in a place where others can see or copy it.
API Key: NUGOYZOUAJIYAJKIAKXZZJNYZAKEZAGUGWEGATQUZTOXZTOŁGEWNCOUDITGNIMEINJILOUGYNUKYONUNZUZ

在 Linux 客户端登录方式如下,作者是在 DGX-1 平台下,Ubuntu 16.04 的系统演示的,如下:



登录成功以后,就可以进行 Lammps 容器下载,如下:

Repositories ∨ nvidia ∨ nvidia/k8s	hpc/lammps docker pull nvcr.io/hpc/lammps:patch23Oct2017	
hpc bigdft candle chroma gamess gromacs lammps lattice-microbes	See here for a document describing prerequisites and setup steps for all HPC containers. See here for a document describing the steps to pull NGC containers. 1. Running LAMMPS	
milc namd pgi-compilers picongpu relion	 There are two options to run the LAMMPS container. You can run LAMMPS in detached mode from the nvidia-docker run command You can start the container in interactive mode and run LAMMPS interactively within the container 	

将命令"docker pull nvcr.io/hpc/lammps:patch23Oct2017"复制到 Linux terminal:



启动 docker 并运行 Lammps

使用 nvidia-docker 命令查看 Lammps 容器镜像信息。

nvcr.io/nvidia/pytorch	18.07-py3	601a2ac5a8a4	3 months ago	5.67GB
paddlepaddle/paddle	0.14.0-gpu-cuda9.0-cudnn7	73383007fb3f	3 months ago	3.21GB
nvcr.io/nvidia/digits	18.07	d5a0acdbdbcf	3 months ago	7.16GB
nvcr.io/nvidia/tensorflow	18.07-py3	8289b0a3b285	3 months ago	3.34GB
nvcr.io/nvidia/tensorflow	18.07-py2	0786194b1dc2	3 months ago	3.34GB
nvcr.io/nvidia/tensorrt	18.07-py3	5c33a10c8b7e	3 months ago	2.61GB
nvcr.io/nvidia/caffe	18.07-py2	69bac2995799	3 months ago	4.29GB
<pre>nvcr.io/openacc/cuda9.0/openmpi2.1.2</pre>	1.0	cceea74653c1	3 months ago	8.73GB
nvcr.io/nvidia/cuda	9.0-cudnn7.1-devel-ubuntu16.04	55a7183596e0	3 months ago	1.65GB
nvcr.io/nvidia/mxnet	18.06-py2	1ee190602e8e	4 months ago	3.89GB
nvcr.io/nvidia/tensorflow	18.06-py3	0d13c9061269	4 months ago	3.4GB
nvidia/cuda	9.0-cudnn7-devel-ubuntu16.04	222eeacc095c	4 months ago	2.61GB
nvcr.io/nvidia/cuda	9.2-base-ubuntu16.04	30a02dc72ed6	4 months ago	132MB
nvcr.io/nvidia/pytorch	18.06-py3	fda0dadff8f3	4 months ago	5.67GB
nvcr.io/nvidia/caffe	18.06-py2	7a48cd22fde1	4 months ago	3.14GB
nvcr.io/nvidia/tensorflow	<none></none>	ba9aba037907	5 months ago	3.82GB
nvcr.io/nvidia/tensorflow	18.05-py3	e878e4ef1885	5 months ago	3.84GB
nvidia-dgx-fw-0102-20180424	latest	e21986a13542	5 months ago	348MB
nvcr.io/nvidia/tensorrt	18.03-py2	2d6903917375	7 months ago	4.33GB
nvcr.io/nvidia/tensorflow	18.03-py3	c0f4402aa77b	7 months ago	2.83GB
nvcr.io/nvidia/cuda	9.0-cudnn7-devel-ubuntu16.04	56240d7febea	8 months ago	1.77GB
nvcr.io/nvidia/tensorrt	18.02-py2	23f146705293	8 months ago	4.31GB
nvcr.io/nvidia/tensorflow	17.12	19afd620fc8e	10 months ago	2.88GB
nvcr.io/hpc/relion	2.1.b1	992ec69c15b4	11 months ago	2.84GB
nvcr.io/hpc/lammps	patch230ct2017	d8ae9ee6bf06	11 months ago	3.48GB
dgxsa@dgx2:~\$				

启动 docker 镜像:

```
dgxsa@dgx1:~$ nvidia-docker run --name MyLammps -v
/home/dgxsa/chengyi/share:/data -it nvcr.io/hpc/lammps: patch230ct2017
/bin/bash
```

docker run Options

- → -i -t or -it: 交互式, 连接到一个"tty"
- → --name: 给容器命名
- → -v /home/dgxsa/chengyi/share:/data : 将 host 主机的/home/dgxsa/chengyi/myshare 存储目

录映射到容器的 data 目录。

<pre><sa@dgx2:~\$ -it="" -v="" chengyi="" data="" dgxsa="" home="" hpc="" lammps:pa<="" mylammps="" nvcr.io="" nvidia-docker="" pre="" runrmname="" share:=""></sa@dgx2:~\$></pre>	itch23
2017 /bin/bash	
<pre>bt@7b2dfc28da4e:/workspace# ls</pre>	
pt@7b2dfc28da4e:/workspace# ll /opt/lammps-patch 230ct2017/	
tal 112	
vxrwxr-x 1 root root 4096 Oct 23 2017 ./	
vxr-xr-x 1 root root 4096 Nov 11 2017/	
vxrwxr-x 2 root root 4096 Oct 23 2017 .github/	
v-rw-r 1 root root 311 Oct 23 2017 .gitignore	
v-rw-r 1 root root 17775 Oct 23 2017 LICENSE	
v-rw-r 1 root root 1690 Oct 23 2017 README	
vxrwxr-x 5 root root 4096 Oct 23 2017 bench/	
vxrwxr-x 5 root root 4096 Oct 23 2017 cmake/	
vxrwxr-x 4 root root 4096 Oct 23 2017 doc/	
vxrwxr-x 61 root root 4096 Oct 23 2017 examples/	
vxrwxr-x 1 root root 4096 Oct 23 2017 lib/	
vxrwxr-x 2 root root 4096 Oct 23 2017 potentials/	
vxrwxr-x 3 root root 4096 Oct 23 2017 python/	
vxrwxr-x 1 root root 40960 Nov 11 2017 src/	
vxrwxr-x 28 root root 4096 Oct 23 2017 tools /	
ot@7b2dfc28da4e:/workspace#	

启动容器以后,就可以像在一台 Linux 服务器上操作了,里面已经配置好了所有运行环境,如果 需要安装其他软件,可以使用命令:

\$ sudo apt-get update

\$ sudo apt-get install xxxx 进行安装。

如果想退出容器,可以使用命令:Ctrl+D

如果想删除容器,可以使用命令: nvidia-docker rm 7b2dfc28da4e; 7b2dfc28da4e 为 docker-ID 如果想退出容器登录界面,但保持容器后台运行,可以使用命令: Ctrl+P, 然后 Ctrl+Q

4标准算例

4.1 准备算例

使用的 Lammps 算例 in.lj 文件如下:

3d Lennard-Jones melt # Enable Accel packages on command line variable x index 4 variable y index 4 variable z index 4 variable iterations index 1000 variable xx equal 20*\$x variable yy equal 20*\$y variable zz equal 20*\$z units lj atom style atomic lattice fcc 0.8442 region box block 0 \${xx} 0 \${yy} 0 \${zz} create box 1 box create atoms 1 box 1 1.0 mass velocity all create 1.44 87287 loop geom pair style lj/cut 2.5 pair coeff 1 1 1.0 1.0 2.5 neighbor 0.3 bin neigh modify delay 0 every 20 check no 1 all nve fix \${iterations} run

4.2 手动运行算例

运行命令及结果如下:

```
root@acc60e90047d:/mnt/lammps# mpirun --allow-run-as-root -n 16 --map-by
core /mnt/lammps-22Aug18/src/lmp_kokkos_cuda_mpi -k on g 2 t 2 -sf kk -pk
kokkos binsize 2.8 comm device -v x 16 -v y 8 -v z 8 -v t 10 -in in.lj
```

参数解释:

-kokkos on/off ... : turn KOKKOS mode on or off (-k), g 2 t 2 分别表示 2 块 GPU, 每个进程包

含2个线程。

-package style ... : invoke package command (-pk)
-suffix gpu/intel/opt/omp : style suffix to apply (-sf)

更多的参数可以运行 lmp kokkos cuda mpi -h 查阅。

4.3 PBS 脚本范例

建立提交脚本 lammps.pbs

```
#PBS -N lammps test
```

```
#PBS -1 nodes=2:gpus=3:ppn=16 #2 个节点,每个节点2块GPU,16核CPU
#PBS -1 walltime=10:00:00
#PBS -5 /bin/bash
#PBS -j oe
cd $PBS_0_WORKDIR
NP=`cat $PBS_NODEFILE|wc -1`
NN=`cat $PBS_NODEFILE|uniq -c|wc -1` #节点数
PPN=`cat $PBS_NODEFILE|uniq -c|sed -n '1,1p'|awk '{print $1}'`#每个节点的进
程数,本例中 PPN=4
mpirun -n $NP --map-by core /mnt/lammps-22Aug18/src/lmp_kokkos_cuda_mpi -k
on g 2 t 2 -sf kk -pk kokkos binsize 2.8 comm device -v x 16 -v y 8 -v z 8
-v t 10 -in in.1j
```

提交作业,并查看是否正确运行。

qsub lammps.pbs qstat

4.4 slurm 脚本范例

建立提交脚本 batch.lammps

#!/bin/bash

```
#SBATCH --job-name=lammps_test
#SBATCH --nodes=2 #总的节点数
#SBATCH --ntasks=32 #总的 CPU 核数
#SBATCH --partition=hsw_v100_32g #集群的队列名称
#SBATCH --gres=gpu:2
#SBATCH --time=01:00:00
cd $PBS_O_WORKDIR
LAMMPS=/mnt/lammps-22Aug18/src/lmp_kokkos_cuda_mpi
INPUT=in.lj
mpirun -n $NP --map-by core $LAMMPS -k on g 2 t 2 -sf kk -pk kokkos
binsize 2.8 comm device -v x 16 -v y 8 -v z 8 -v t 10 -in $INPUT >log.out
2>&1
```

关于 lammps 的 slurm 运行脚本中指定 GPU,需要在 slurm.conf 中配置,需要把下面的配置文件 slurm.conf 中关于 generic resource (gres)的配置注释去掉:

[chengyi@psgcluster test]\$ cat /cm/images/compute-image/etc/slurm/slurm.conf]grep GresTypes #GresTypes=gpu