

NVIDIA 英伟达技术培训

摘要：本次技术分享主要介绍 NVIDIA 英伟达 GPU 加速工具 NGC 在生命科学领域的使用及注意事项。

目录：

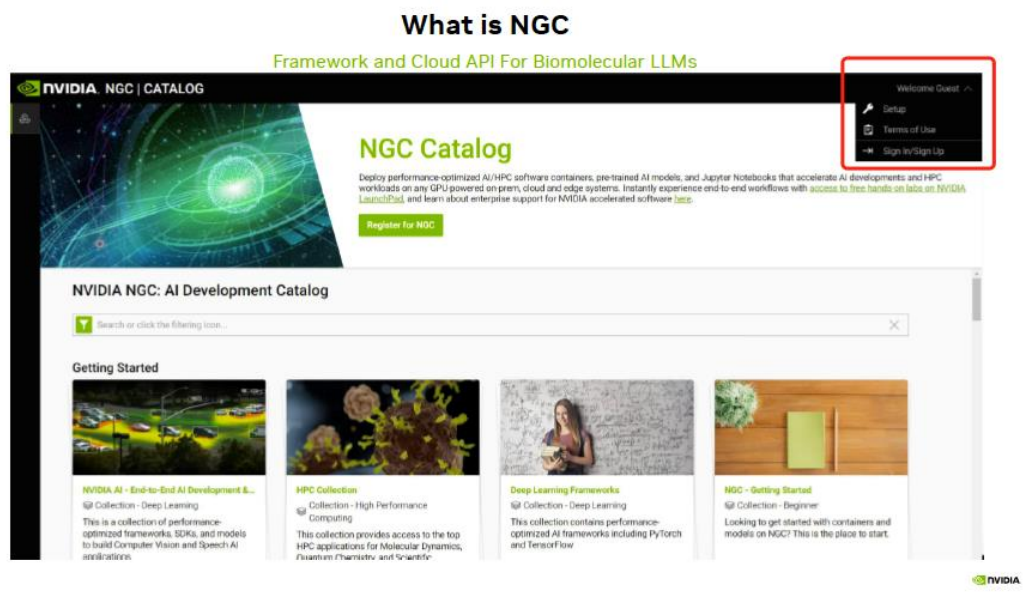
- 一、NGC 介绍
- 二、Clara Discovery
- 三、DL Framework
- 四、BioNeMo
- 五、RAPIDS
- 六、CUDA

一、NGC 介绍

1、什么是 NGC

NGC (NVIDIA GPU Cloud), GPU 优化的 AI 企业服务、软件和支持, 作为面向 AI、机器学习和高性能计算的服务、软件和支持的门户, NVIDIA NGC 提供一系列完全托管的云服务, 包括用于 NLU 和语音 AI 解决方案的 NeMo LLM、BioNemo 和 Riva Studio。AI 从业者可以利用 NVIDIA Base Command 进行模型训练, 利用 NVIDIA Fleet Command 进行模型管理, 并利用 NGC 专用注册表安全共享专有 AI 软件。此外, NGC 还拥有 GPU 优化的 AI 软件、SDK 和 Jupyter Notebook 的目录, 可帮助加速 AI 工作流, 并通过 NVIDIA AI Enterprise 提供支持。

2、注册和登录



网址: <https://catalog.ngc.nvidia.com/>

Register

 **NVIDIA**. NGC

Welcome to NVIDIA NGC - your portal to NVIDIA AI, Omniverse and high-performance computing (HPC).

Enter your email to sign in.

Email Address

Continue

Use alternate method

Register

 **NVIDIA**. NGC

Set Your Profile

First Name

Last Name

Location

Job Role

Organization

Organization URL

Industry Segment(s)

Areas of Interest

☐ I agree to the [NVIDIA GPU Cloud Terms of Use](#)

☐ By obtaining software through NGC, I agree NVIDIA can share my registration details with the software providers who may use my information as permitted by their privacy policies.

Submit

注册

NVIDIA
NGC | CATALOG

CATALOG

Explore Catalog
Collections
Containers
 Helm Charts
 Images
 Repositories

Getting Started

Welcome to NGC and start building your AI now. If you're new to AI, we will help you get started with performance optimized AI framework, deep learning, and more. Instantly experience end-to-end workflows with [access to free hands-on labs on NVIDIA LaunchPad](#), and learn about enterprise accelerated software [here](#).

[Start AI with NGC](#)

Select Organization...
Select Organization...
Account Settings
Setup
Terms of Use
Privacy Policy
Sign Out

NVIDIA NGC: AI Development Catalog

Search or click the filtering icon...

Getting Started

NVIDIA AI - End-to-End AI Development & Deployment
Collection - Deep Learning
This is a collection of performance-optimized frameworks, SDKs, and models to build Computer Vision and Speech AI applications.

[View Labels](#)

HPC Collection
Collection - High Performance Computing
This collection provides access to the top HPC applications for Molecular Dynamics, Quantum Chemistry, and Scientific Computing.

[View Labels](#)

Deep Learning Frameworks
Collection - Deep Learning
This collection contains performance-optimized AI frameworks including PyTorch and TensorFlow.


[View Labels](#)

NGC - Getting Started
Collection - Beginner
Looking to get started with containers and models on NGC? This is the place to start.

[View Labels](#)

登录


Setup



Generate API Key

Generate your own API key in order to use the NGC service through the Docker client or through NGC CLI.

[Get API Key](#)

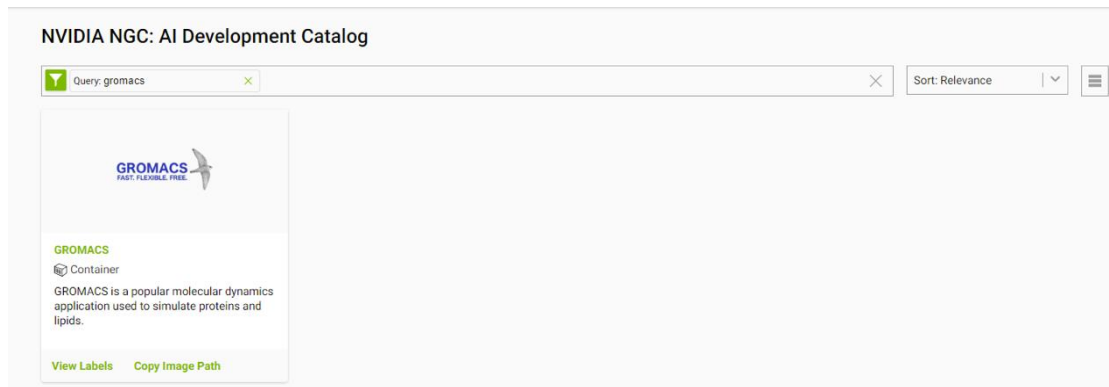


CLI

The NGC command line interface (NGC CLI) can run deep learning jobs on NVIDIA Docker containers.

[Documentation](#)[Downloads](#)

Setup, 生成 API Key, 保存该字符串



搜索常用应用如 GROMACS，拷贝镜像地址：nvcr.io/hpc/gromacs:2022.3

```
(base) rtx@rtxA6000:~$ docker pull nvcr.io/hpc/gromacs:2022.3
2022.3: Pulling from hpc/gromacs
eaead16dc43b: Pull complete
aba94d2c2c3c: Pull complete
7e98827f3a56: Pull complete
11886930ecae: Pull complete
55ac2a638077: Pull complete
bd0a7b9fdf15: Pull complete
e84db3a985d2: Pull complete
c511d6161745: Pull complete
78c571fb0bd9: Pull complete
5e4511652332: Pull complete
d733a4c3b441: Pull complete
7bd1251d6a59: Pull complete
417a646d977b: Pull complete
71b480fa9069: Pull complete
77e75aca301d: Pull complete
5a19b36d69bd: Pull complete
09beb5200b4b: Pull complete
bf41a3abe3d5: Pull complete
04168d8ae009: Pull complete
c546e7e61957: Pull complete
ae8beec5d561: Pull complete
9f5139e6eb66: Pull complete
ac47ebdd005d: Pull complete
42b5719cf197: Pull complete
364b4f61df81: Pull complete
5e32a66a3d53: Pull complete
e8e802167fd7: Pull complete
77454f410724: Pull complete
a1e08c176120: Pull complete
83036606ae38: Pull complete
Digest: sha256:b55e99df2acb691f6dd147d33f7fe3519731fe572599c0bdd9b17718a133d551
Status: Downloaded newer image for nvcr.io/hpc/gromacs:2022.3
nvcr.io/hpc/gromacs:2022.3
```

docker pull 拉取到本地，即可使用

命令：`docker pull nvcr.io/hpc/gromacs:2022.3`

3、Docker 安装

```
$ sudo apt-get update
```

```
$ sudo apt-get install -y nvidia-docker2
```

```
$ sudo systemctl restart docker
```

执行测试：

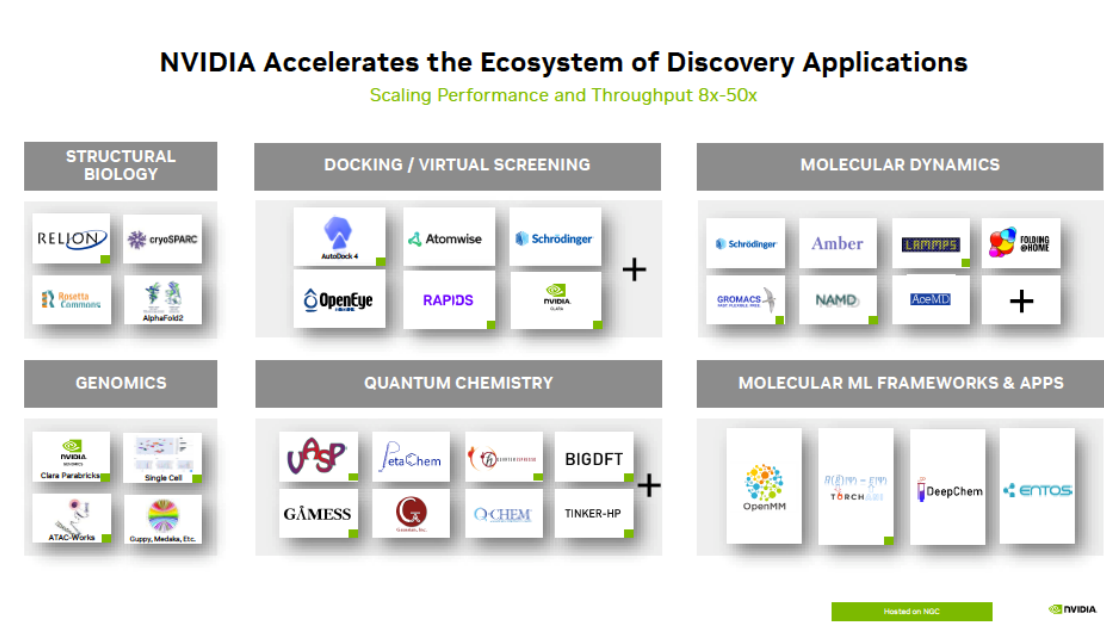
```
$ sudo docker run --rm --gpus all nvidia/cuda:11.0-base nvidia-smi
```

安装成功则显示如下信息：

```
+-----+
| NVIDIA-SMI 450.51.06    Driver Version: 450.51.06    CUDA Version: 11.0    |
+-----+-----+
| GPU  Name      Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|-----+-----+-----+
| 0   Tesla T4      On         | 00000000:00:1E:0 Off  |      0      0      |
| N/A   34C    P8      9W /  70W |  0MiB / 15109MiB |      0%    Default  |
+-----+-----+-----+

+-----+
| Processes:                                                       GPU Memory |
|  GPU   GI    CI          PID    Type   Process name                  Usage      |
|-----+-----+
| No running processes found                                     |
+-----+
```

4、NGC 生态体系



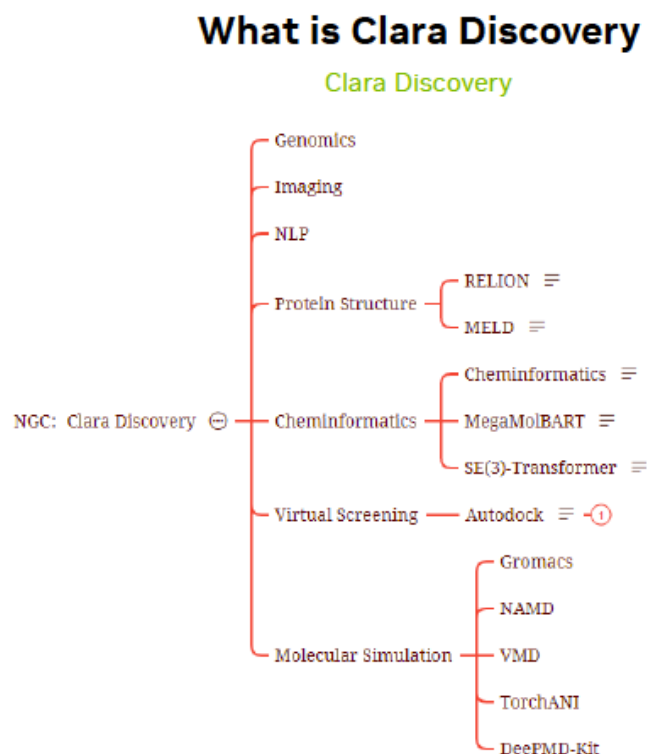
英伟达在生科领域有一整套的生态体系，如结构生物学、虚拟筛选、分子动力学、基因、量子化学、分子机器学习框架等。

二、NVIDIA Clara Discovery

1、什么是 Clara Discovery

NVIDIA Clara Discovery 集 GPU 加速及优化的框架、工具、应用和预训练模型于一体，用于计算药物研发。Clara Discovery 专为支持跨学科工作流而构建，可帮助科学家和研究人员更快地将药物投放市场，并为疾病机制研究提供新的可能性。

2、Clara Discovery 的能力

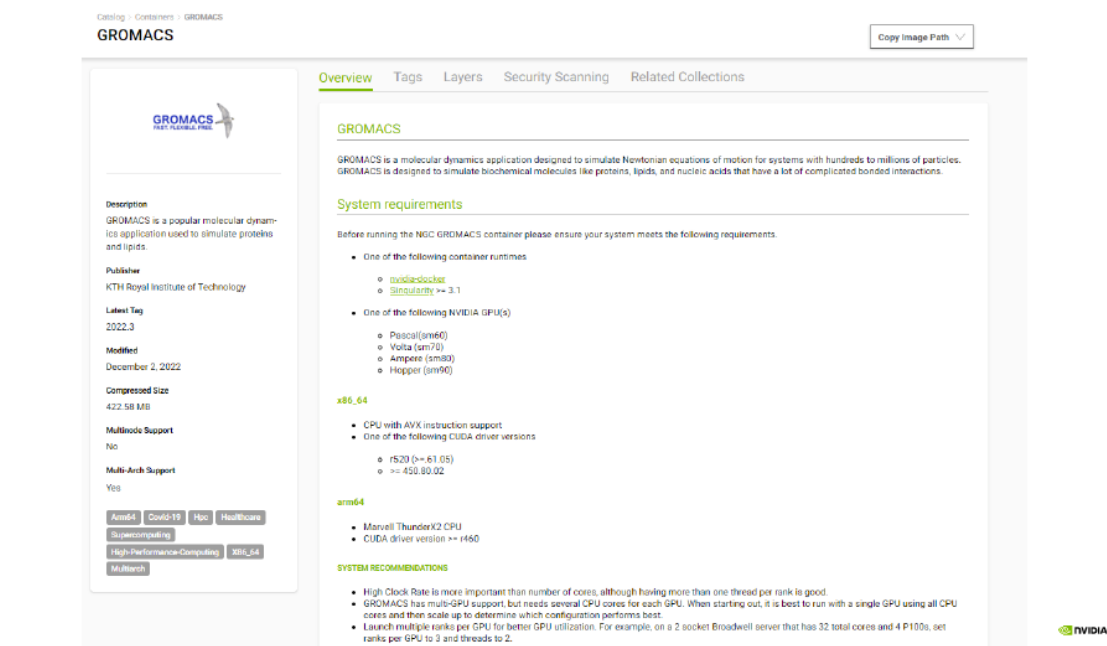


其中用于生物大分子蛋白质、核酸、小分子药物等的分子动力学模拟软件主要有 GROMACS、Amber、LAMMPS、NAMD 等。



对于不同的模拟体系选择的模拟软件包也有差异，如蛋白质体系可以选择 GROMACS、Amber、NAMD；对于 DNA、RNA 体系则首选 Amber；材料体系会选择 LAMMPS。

3、GROMACS



a. 运行使用

如前所述，可以通过 `docker pull` 将 GROMACS 拉取到本地使用，同时根据说明安装相应驱动，即可运行使用。

Running with `nvidia-docker`

WITHOUT INFINIBAND

```
DOCKER="nvidia-docker run -it --rm -v ${PWD}:/host_pwd --workdir /host_pwd nvr.io/hpc/gromacs:${GROMACS_TAG}"
```

WITH INFINIBAND

```
DOCKER="nvidia-docker run -it --rm -v ${PWD}:/host_pwd --workdir /host_pwd --device=/dev/infiniband --cap-add=IPC_LOCK --net=host nvr.io/hpc/gromacs:${GROMACS_TAG}"
```

Prepare the benchmark data

```
${DOCKER} gmx grompp -f pme.mdp
```

Run GROMACS

```
${DOCKER} gmx mdrun -ntmpi 4 -nb gpu -pin on -v -noconfout -nsteps
```

```
5000 -s -ntomp 10 topol.tpr
```

b. 源码安装

对于单机多卡，以上环境基本可以满足运行。但对于需要多节点运行的时候，建议采用源码安装方式。

1) 编译器：由于需要完全的 C++17 支持，编译器版本最低要求如下：

- GNU (gcc libstdc++) 7
- Intel (icc) 19.1
- LLVM (clang/libc++) 5
- Microsoft (MSVC) 2017 15.7

2) 并行：支持多种并行方式

- 内部 thread mpi 并行（进程级别并行），效率高，默认支持，无法跨节点并行：gmx mdrun ntmpi 4
- openmp 并行（线程级别并行），默认支持，无法跨节点并行：gmx mdrun ntomp
- 外部的 mpi 并行（进程级别并行），需要安装 MPI 软件，可以跨节点运行：mpirun np 4 gmx_mpi mdrun

注：1 和 2 可以同时使用，2 和 3 可以同时使用，1 和 3 不能同时使用。在一个节点内，1 比 3 性能更好。

- 单节点运行时，通常采用 1+2 或者只用 1 的方式运行：gmx mdrun ntmpi 4 ntomp 6

- 跨节点运行时，通常采用 2+3 的方式运行：mpirun np 2 gmx_mpi mdrun ntomp 6

3) 源码安装

```
wget http://www.fftw.org/fftw-3.3.9.tar.gz
tar xzf fftw-3.3.9.tar.gz
cd fftw-3.3.9
./configure prefix=/raid/daizx/HPC/fftw-3.3.9/install --enable-sse2 --enable-avx
--enable-avx2 --enable-float --enable-shared
make
make install

tar xzf gromacs-2021.2.tar.gz
cd gromacs-2021.2
mkdir build
cd build
```

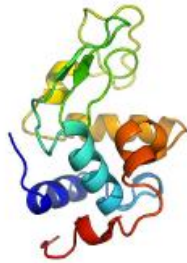


```
cmake ..-DCMAKE_INSTALL_PREFIX=/raid/daizx/HPC/gromacs-2021.2/install
-DGMX_FFT_LIBRARY=fftw3      -DCMAKE_PREFIX_PATH=/raid/daizx/HPC/fftw-
3.3.9/install -
DGMX_GPU=CUDA-DCUDA_TOOLKIT_ROOT_DIR=/usr/cuda
make j8
make check
make install
```

注：安装 mpi 时，-DCMAKE_C_COMPILER= mpicc -DCMAKE_CXX_COMPILER=mpicxx -
DGMX_MPI=on

c. 模拟流程

以模拟水中的溶菌酶为例，详细介绍 GROMACS 模拟的具体步骤。



步骤：准备拓扑-定义盒子和添加溶剂-添加离子-能量最小化-NVT 平衡-NPT 平衡-MD 模拟-分析

演示网址：<http://www.mdtutorials.com/gmx/index.html>

1) 准备拓扑

- 打开 RCSB 网站，输入鸡蛋白溶菌酶的 PDB 编号下载；

RCSB PDB 180953 Biological Macromolecular Structures Enabling Breakthroughs in Research and Education

1AKI

THE STRUCTURE OF THE ORTHORHOMBIC FORM OF HEN EGG-WHITE LYSOZYME AT 1.5 ANGSTROMS RESOLUTION

DOI: 10.2210/pdb1AKI/pdb

Classification: **HYDROLASE**

Organism(s): *Gallus gallus*

Mutation(s): No

Deposited: 1997-05-19 Released: 1997-11-19

Deposition Author(s): Carter, D., He, J., Ruble, J.R., Wright, B.

Experimental Data Snapshot

Method: X-RAY DIFFRACTION

Resolution: 1.50 Å

R-Value Work: 0.212

wwPDB Validation

Metric	Percentile Ranks	Value
Clashscore	6	6
Ramachandran outliers	0	0
Sidechain outliers	4.8%	4.8%

RCSB 网站 <https://www.rcsb.org/>

- 去掉晶体结构中的结晶水；

```
grep -v HOH laki.pdb > 1AKI_clean.pdb
```

- 使用 `pdb2gmx` 模块，生成分子拓扑文件、位置限制文件和后处理结构文件。

```
gmx pdb2gmx -f 1AKI_clean.pdb -o 1AKI_processed.gro -water spce
```

2) 定义盒子和添加溶剂

- 使用 `editconf` 模块定义盒子的尺寸

```
gmx editconf -f 1AKI_processed.gro -o 1AKI_newbox.gro -c -d 1.0 -bt cubic
```

注：-c:将蛋白质置于盒子的中心；-d 1.0: 到盒子边缘的距离至少为 1.0 nm；-bt cubic: 盒子类型是立方体。

- 使用 `solvate` 模块向盒子中填充水

```
gmx solvate -cp 1AKI_newbox.gro -cs spc216.gro -o 1AKI_solv.gro -p topol.top
```

注：-cp:editconf 步骤中的输出文件；-cs:溶剂的构型文件来自标准的 GROMACS, spc216.gro 是通用的已平衡的三位点溶剂模型；-o:输出文件名；-p:指定拓扑文件的名称。

3) 添加离子

由于生命体系中不存在净电荷，所以必须添加离子，以保证总电荷为零。

- 在添加离子前，需要生成包含了所有原子的所有参数 `tpx` 文件。使用 `grompp` 生成。

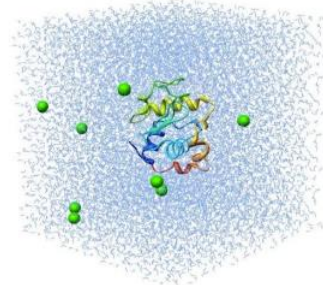
```
gmx grompp -f ions.mdp -c 1AKI_solv.gro -p topol.top -o ions.tpx
```

- 使用 genion 添加离子

```
gmx genion -s ions.tpr -o lAKI_solv_ions.gro -p topol.top -pname NA -
nname CL -neutral
```

```
cat ions.mdp (原子描述)
; ions.mdp - used as input into grompp to generate ions.tpr
; Parameters describing what to do, when to stop and what to save
integrator = steep      ; Algorithm (steep = steepest descent minimization)
emtol      = 1000.0     ; Stop minimization when the maximum force < 1000.0 kJ/mol/nm
emstep     = 0.01      ; Minimization step size
nsteps     = 50000      ; Maximum number of (minimization) steps to perform

; Parameters describing how to find the neighbors of each atom and how to calculate the interactions
nstlist    = 1         ; Frequency to update the neighbor list and long range forces
cutoff-scheme = Verlet  ; Buffered neighbor searching
ns_type     = grid     ; Method to determine neighbor list (simple, grid)
coulombtype = cutoff    ; Treatment of long range electrostatic interactions
rcoulomb    = 1.0      ; Short-range electrostatic cut-off
rvdw        = 1.0      ; Short-range Van der Waals cut-off
pbc         = xyz      ; Periodic Boundary Conditions in all 3 dimensions
```



4) 能量最小化

在开始动力学模拟之前，必须保证体系的结构正常，原子之间的距离不会过近，几何构型合理。这个过程称为能量最小化 (EM, energy minimization)。

- 能量最小化过程与添加离子过程类似，再次使用 grompp 将结构、拓扑和模拟参数 (.mdp 文件) 写入一个能量最小化的输入文件 (.tpr)。

```
gmx grompp -f minim.mdp -c lAKI_solv_ions.gro -p topol.top -o em.tpr
```

- 调用 mdrun 来进行能量最小化了：

```
gmx mdrun -deffnm em
```

-deffnm 选项定义了输入文件和输出文件的名称。

5) 平衡

平衡往往分两个阶段进行。需要将体系置于设定的模拟温度下，以确定溶质(蛋白质)的合理取向。达到正确的温度(基于动能)之后，要对体系施加压力直到它达到合适的密度。

- 第一个阶段在 NVT 系综(粒子数，体积和温度都是恒定的)下进行，这个系综也被称为等温等容系综或正则系综。

```
gmx grompp -f nvt.mdp -c em.gro -r em.gro -p topol.top -o nvt.tpr
```

```
gmx mdrun -deffnm nvt -ntmpi 4 -ntomp 6 -nb gpu -bonded gpu -pme gpu
-npme 1
```

- 第二个阶段是在 NPT 系综下进行的，其中粒子数，压力和温度都保持不变。这个系综也被称为等温等压系综，最接近实验条件。

```
gmx grompp -f npt.mdp -c nvt.gro -r nvt.gro -t nvt.cpt -p topol.top -o
npt.tpr
```

```
gmx mdrun -deffnm npt -ntmpi 4 -ntomp 6 -nb gpu -bonded gpu -pme gpu
```

-npme 1

注: -ntmpi 4 , 启动 4 个 thread-mpi 进程; -ntomp 6 , 每个进程采用 6 个线程; -nb gpu -bonded gpu -pme gpu, 把 3 种力的计算都放到 GPU 上进行; -npme 1 , 单独分配一块 GPU 计算 PME.

*NVT 参数文件:

```
cat nvt.mdp
title = OPLS Lysozyme NVT equilibration
define = -DPOSRES ; position restrain the protein
; Run parameters
integrator = md ; leap -frog integrator
nsteps = 50000 ; 2 * 50000 = 100 ps
dt = 0.002 ; 2 fs
; Output control
nstxout = 500 ; save coordinates every 1.0 ps
nstvout = 500 ; save velocities every 1.0 ps
nstenergy = 500 ; save energies every 1.0 ps
nstlog = 500 ; update log file every 1.0 ps
; Bond parameters
continuation = no ; first dynamics run
constraint_algorithm = lincs ; holonomic constraints
constraints = h -bonds ; bonds involving H are constrained
lincs_iter = 1 ; accuracy of LINCS
lincs_order = 4 ; also related to accuracy
; Nonbonded settings
cutoff-scheme = Verlet ; Buffer neighbor searching
ns_type = grid ; search neighboring grid cells
nstlist = 10 ; 20 fs, largely irrelevant with Verlet
rcoulomb = 1.0 ; short -range electrostatic cutoff (in nm)
rvdw = 1.0 ; short -range van der Waals cutoff (in nm)
DispCorr = EnerPres ; account for cut-off vdW scheme
; Electrostatics
coulombtype = PME ; Particle Mesh Ewald for long -range electrostatics
pme_order = 4 ; cubic interpolation
fourierspacing = 0.16 ; grid spacing for FFT
; Temperature coupling is on
tcoupl = V -rescale ; modified Berendsen thermostat
tc-grps = Protein Non-Protein ; two coupling groups - more accurate
tau_t = 0.1 0.1 ; time constant, in ps
ref_t = 300 300 ; reference temperature, one for each group, in
K
; Pressure coupling is off
```

```

pcoupl = no ; no pressure coupling in NVT
; Periodic boundary conditions
pbc = xyz ; 3 -D PBC
; Velocity generation
gen_vel = yes ; assign velocities from Maxwell distribution

```

*NPT 参数文件:

```

cat npt.mdp
title = OPLS Lysozyme NPT equilibration
define = -DPOSRES ; position restrain the protein
; Run parameters
integrator = md ; leap -frog integrator
nsteps = 50000 ; 2 * 50000 = 100 ps
dt = 0.002 ; 2 fs
; Output control
nstxout = 500 ; save coordinates every 1.0 ps
nstvout = 500 ; save velocities every 1.0 ps
nstenergy = 500 ; save energies every 1.0 ps
nstlog = 500 ; update log file every 1.0 ps
; Bond parameters
continuation = yes ; Restarting after NVT
constraint_algorithm = lincs ; holonomic constraints
constraints = h -bonds ; bonds involving H are constrained
lincs_iter = 1 ; accuracy of LINCS
lincs_order = 4 ; also related to accuracy
; Nonbonded settings
cutoff-scheme = Verlet ; Buffered neighbor searching
ns_type = grid ; search neighboring grid cells
nstlist = 10 ; 20 fs, largely irrelevant with Verlet scheme
rcoulomb = 1.0 ; short -range electrostatic cutoff (in nm)
rvdw = 1.0 ; short -range van der Waals cutoff (in nm)
DispCorr = EnerPres ; account for cut -off vdW scheme
; Electrostatics
coulombtype = PME ; Particle Mesh Ewald for long -range electrostatics
pme_order = 4 ; cubic interpolation
fourierspacing = 0.16 ; grid spacing for FFT
; Temperature coupling is on
tcoupl = V -rescale ; modified Berendsen thermostat
tc-grps = Protein Non -Protein ; two coupling groups - more accurate
tau_t = 0.1 0.1 ; time constant, in ps
ref_t = 300 300 ; reference temperature, one for each group, in
K
; Pressure coupling is on

```

```

pcoupl = Parrinello -Rahman ; Pressure coupling on in NPT
pcoupltype = isotropic ; uniform scaling of box vectors
tau_p = 2.0 ; time constant, in ps
ref_p = 1.0 ; reference pressure, in bar
compressibility = 4.5e -5 ; isothermal compressibility of water, bar-1

```

6) MD 模拟

- 随着两个平衡阶段的完成，体系已经在需要的温度和压强下平衡好，开始进行 MD 模拟。

```

gmx grompp -f md.mdp -c npt.gro -t npt.cpt -p topol.top -o md_0_1.tpr
gmx mdrun -deffnm md_0_1 -ntmpi 4 -ntomp 6 -nb gpu -bonded gpu -pme
gpu -npme 1

```

*md 参数文件

```

cat md.mdp
title = OPLS Lysozyme NPT equilibration
; Run parameters
integrator = md ; leap -frog integrator
nsteps = 500000 ; 2 * 500000 = 1000 ps (1 ns)
dt = 0.002 ; 2 fs
; Output control
nstxout = 0 ; suppress bulky .trr file by specifying
nstvout = 0 ; 0 for output frequency of nstxout,
nstfout = 0 ; nstfout, and nstfout
nstenergy = 5000 ; save energies every 10.0 ps
nstlog = 5000 ; update log file every 10.0 ps
nstxout-compressed = 5000 ; save compressed coordinates every 10.0 ps
compressed-x-grps = System ; save the whole system
; Bond parameters
continuation = yes ; Restarting after NPT
constraint_algorithm = lincs ; holonomic constraints
constraints = h -bonds ; bonds involving H are constrained
lincs_iter = 1 ; accuracy of LINCS
lincs_order = 4 ; also related to accuracy
; Neighborsearching
cutoff-scheme = Verlet ; Buffered neighbor searching
ns_type = grid ; search neighbor ing grid cells
nstlist = 10 ; 20 fs, largely irrelevant with Verlet scheme
rcoulomb = 1.0 ; short -range electrostatic cutoff (in nm)
rvdw = 1.0 ; short -range van der Waals cutoff (in nm)
; Electrostatics
coulombtype = PME ; Particle Mesh Ewald for long -range electrostatics

```

```

pme_order = 4 ; cubic interpolation
fourierspacing = 0.16 ; grid spacing for FFT
; Temperature coupling is on
tcoupl = V -rescale ; modified Berendsen thermostat
tc-grps = Protein Non -Protein ; two coupling groups - more accurate
tau_t = 0.1 0.1 ; time constant, in ps
ref_t = 300 300 ; reference temperature, one for each group, in
K
; Pressure coupling is on
pcoupl = Parrinello -Rahman ; Pressure coupling on in NPT
pcoupltype = isotropic ; uniform scaling of box vectors
tau_p = 2.0 ; time constant, in ps
ref_p = 1.0 ; reference pressure, in bar

```

执行结果:

```

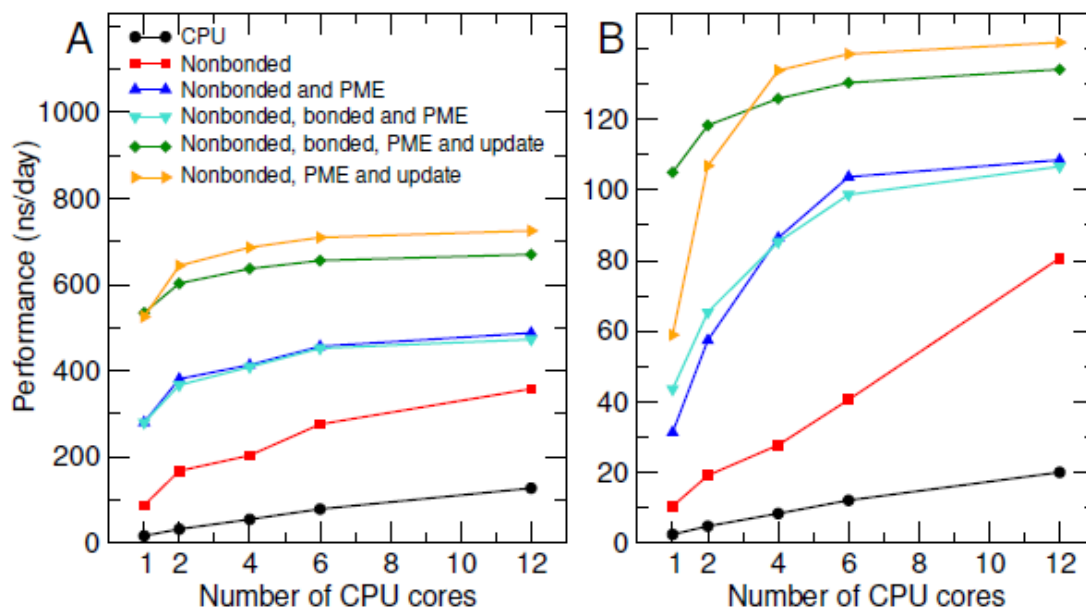
Dynamic load balancing report:
DLB was off during the run due to low measured imbalance.
Average load imbalance: 2.5%.
The balanceable part of the MD step is 68%, load imbalance is computed
from this.
Part of the total run time spent waiting due to load imbalance: 1.7%.
Average PME mesh/force load: 0.925
Part of the total run time spent waiting due to PP/PME imbalance: 0.7 %
  Core t (s) Wall t (s) (%)
Time: 5437.730 226.574 2400.0
      (ns/day) (hour/ns)
Performance: 381.333 0.063
GROMACS reminds you: "We Don't Bother Anyone" (LIVE)

```

其中值 381.333ns/day, 该值越大越好。

7) Offload 方式

耗时部分包含:NB、BF、PME 和 Update & constraints。各部分可以选择性 offload 到 GPU。采用 RNase (A 24k 原子) 和 GluC1 (B: 142k 原子), 测试不同 CPU 核心和不同部分卸载的性能。



- Offload NB, 性能提升明显。增加 CPU 核心，性能继续提升。
- Offload PME, 性能提升明显。当 CPU 核心从 6 增大到 12 时，性能基本不变。
- Offload BF, 性能差异不大。
- Offload Update & constraints, 减少 CPU 和 GPU 之间的数据交换，性能进一步提升。
- 当 BF 重新回到 CPU, CPU 和 GPU 之间的负载会出现平衡点。平衡点取决于系统和算例大小。

8) 优化方法

2020 版本的几个特点，默认没有 enabled。需要在运行时添加以下参数：

- To enable the update and constraints part of the timestep
export GMX_FORCE_UPDATE_DEFAULT_GPU=true
- For communications between PME and PP tasks
export GMX_GPU_PME_PP_COMMS=true
- For halo exchange communications between PP tasks
export GMX_GPU_DD_COMMS=true

对于单 GPU 来说，只需要第一个参数。

特别说明，采用 2 个 GPU 时，也需要分配 4 个进程，其中 1 个用来计算 PME，注意 PP tasks 和 PME tasks 的平衡。

9) 测试

硬件：DGX A100

- 1GPU: `gmx mdrun -s ./topol.tpr -ntomp 10 -nb gpu -bonded gpu -pme gpu -nstlist 400 -v -nsteps 100000 -resetstep 90000 -noconfout`
- 2GPU: `CUDA_VISIBLE_DEVICES=0,1 gmx mdrun -s ./topol.tpr -ntmpi 4 -ntomp 10 -nb gpu -bonded gpu -pme gpu -npme 1 -nstlist 400 -v -nsteps 100000 -resetstep 90000 -noconfout`
- 4GPU: `gmx mdrun -s ./topol.tpr -ntmpi 4 -ntomp 10 -nb gpu -bonded gpu -pme gpu -npme 1 -nstlist 400 -v -nsteps 100000 -resetstep 90000 -noconfout`

注：-ntmpi 4，启动 4 个 mpi 进程；-ntomp 10，每个进程采用 10 个线程；-nb gpu -bonded gpu -pme gpu，把 3 种力的计算都放到 GPU 上进行；-npme 1，单独分配一块 GPU 计算 PME；-nstlist 400，计算 400 步更新一次原子表，通过实验证明，400 能获得最好的性能，也不会带来精度损失；-nsteps 100000，计算 100000 步；-resetstep 90000，在 90000 步进行 reset，避免初始化带来的速度误差；-noconfout，计算完成时不要写输出文件，避免带入 IO 问题。默认时间步为 2fs。

10) 性能

官网测试结果：<https://developer.nvidia.com/hpc-application-performance>

Application	Metric	Test Modules	Bigger is better	Dual Cascade Lake 6240 (CPU-Only)	1x A100 SXM 80GB	2x A100 SXM 80GB	4x A100 SXM 80GB	8x A100 SXM 80GB	1x A100 PCIe 80GB	2x A100 PCIe 80GB	4x A100 PCIe 80GB	8x A100 PCIe 80GB
GROMACS [ADH Dodec]	ns/day	ADH Dodec	yes	56	329	-	477	-	327	-	494	-
GROMACS [ADH Dodec]	NRF	ADH Dodec	yes	1x	8x	-	11x	-	8x	-	12x	-
GROMACS [Cellulose]	ns/day	Cellulose	yes	16	96	139	226	248	96	130	169	187
GROMACS [Cellulose]	NRF	Cellulose	yes	1x	9x	13x	21x	23x	9x	12x	16x	18x
GROMACS [STMV]	ns/day	STMV	yes	4	22	38	56	112	22	38	54	77
GROMACS [STMV]	NRF	STMV	yes	1x	6x	11x	16x	31x	6x	11x	15x	22x

DGX STATION A100 测试结果：

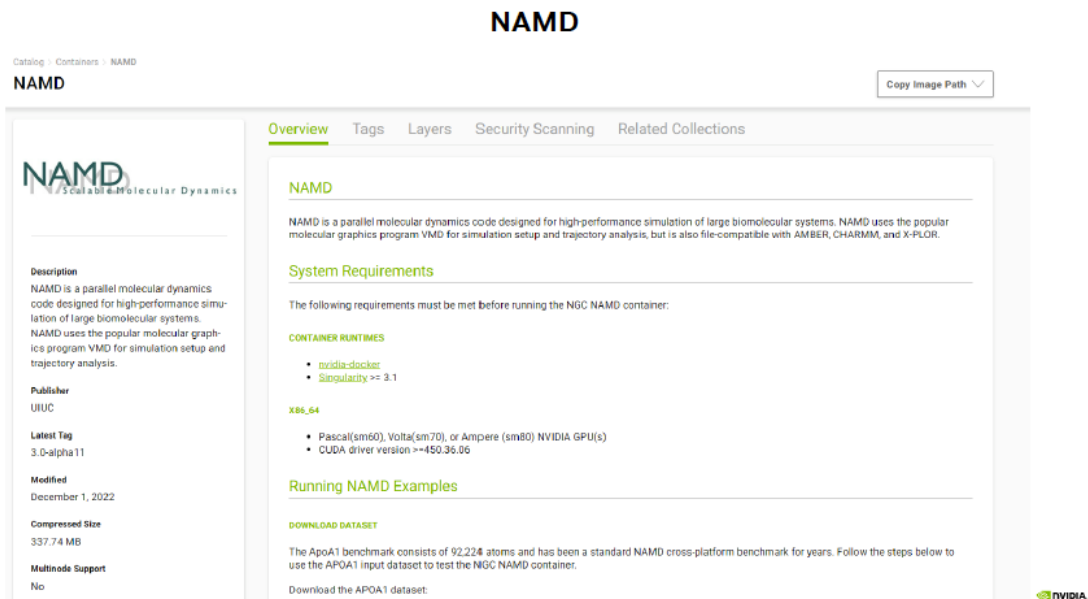
Case	Method	1GPU	2GPU	4GPU
ADH dodec	Default	162	175	194
	Optimize	291	431	569
Cellulose	Default	47	58	69
	Optimize	87	154	224
STMV	Default	12	15	20
	Optimize	20	32	55

- 优化后性能提升明显，多 GPU 提升都在 2 倍以上。
- 算例规模越大，扩展性越好
- 采用 8 块 GPU 时，nvlink 性能提升 40%左右。

拓展链接：运行 GROMACS 进行分子动力学模拟

<https://www.alibabacloud.com/help/zh/e-hpc/latest/app-gromacs>

4、NAMD



和 GROMACS 一样，可以通过 docker pull 将 NAMD 拉取到本地使用。

NGC supports the Docker runtime through the nvidia-docker plugin

- WITHOUT INFINIBAND

```
DOCKER="nvidia-docker run -it --rm -v $(pwd):/host_pwd
nvcr.io/hpc/namd:${NAMD_TAG}"
```

- WITH INFINIBAND

```
DOCKER="nvidia-docker run -it --rm -v $(pwd):/host_pwd --
device=/dev/infiniband --cap-add=IPC_LOCK --net=host
nvcr.io/hpc/namd:${NAMD_TAG}"
```

- Launch NAMD across all CPU cores, utilizing all GPUs, on your local machine or single node:

```
${DOCKER} ${NAMD_EXE} +ppn $(nproc) +setcpuaffinity +idlepoll {input
file}
```

The `nproc` command is used to specify all available CPU cores should be used. Depending on system setup manually specifying the number of PE's may yield better performance.

An example shell script demonstrating this mode is available for 3.0-alpha11 tag.


5、LAMMPS

LAMMPS

Catalog > Containers > LAMMPS

LAMMPS Copy Image Path

Overview Tags Layers Security Scanning Related Collections



Description
Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) is a software application designed for molecular dynamics simulations.

Publisher
Sandia National Lab

Latest Tag
patch_4May2022

Modified
December 1, 2022

Compressed Size
514.02 MB

MultiNode Support

LAMMPS

Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) is a software application designed for molecular dynamics simulations. It has the potentials for solid-state materials (metals, semiconductor), soft matter (biomolecules, polymers), and coarse-grained or mesoscopic systems. The main use case is atom scale particle modeling or, more generically, as a parallel particle simulator at the atomic, meso, or continuum scale. LAMMPS runs on single processors or in parallel using message-passing techniques and a spatial-decomposition of the simulation domain. Read more on the [LAMMPS website](#).

System requirements

Before running the NGC LAMMPS container please ensure your system meets the following requirements.

- One of the following container runtimes
 - [nvidia-docker](#)
 - [Singularity](#) >= 3.1
- One of the following NVIDIA GPU(s)
 - Pascal(sm60)
 - Volta (sm70)
 - Ampere (sm80)

x86_64

- CPU with AVX2 instruction support

- Running with `nvidia-docker`

```
cd $BENCHMARK DIR
```

```
docker run --rm --gpus all --ipc=host -v $PWD:/host_pwd -w /host_pwd  
nvcr.io/hpc/lammps:DDMMYYYYY ./run_lammps.sh
```

- Running multi-node with Slurm and Singularity

Clusters running the Slurm resource manager and Singularity container runtime may launch parallel LAMMPS experiments directly through `srun`. The NGC LAMMPS container supports `pmi2`, which is available within most Slurm installations, as well as `pmix3`. A typical parallel experiment would take the following form.

```
srun --mpi=pmi2 [srun_flags] singularity run --nv [singularity_flags]  
lmp [lammps_flags]
```

拓展链接：使用 LAMMPS 软件进行高性能计

算, https://help.aliyun.com/document_detail/57898.html

三、DL Framework

除了医药领域，目前各行各业都会用到深度学习框架。同样在 NGC 上可以通过搜

索 PYTORCH、Tensorflow 得到相关镜像。

Catalog > Containers > PyTorch

PyTorch

Copy Image Path

Deploy to Vertex AI

Overview

Tags

Layers

Security Scanning

Related Collections

PyTorch

Accelerated with NVIDIA

Description

PyTorch is a GPU accelerated tensor computational framework. Functionality can be extended with common Python libraries such as NumPy and SciPy. Automatic differentiation is done with a tape-based system at the functional and neural network layer levels.

Publisher

Facebook

Latest Tag

22.11-py3

Modified

December 1, 2022

Compressed Size

8.07 GB

PyTorch

PyTorch is an optimized tensor library for deep learning using GPUs and CPUs. Automatic differentiation is done with a tape-based system at both a functional and neural network layer level. This functionality brings a high level of flexibility and speed as a deep learning framework and provides accelerated NumPy-like functionality. NGC Containers are the easiest way to get started with PyTorch. The PyTorch NGC Container comes with all dependencies included, providing an easy place to start developing common applications, such as conversational AI, natural language processing (NLP), recommenders, and computer vision.

The PyTorch NGC Container is optimized for GPU acceleration, and contains a validated set of libraries that enable and optimize GPU performance. This container also contains software for accelerating ETL ([DALI](#), [RAPIDS](#)), Training ([cuDNN](#), [NCCL](#)), and Inference ([TensorRT](#)) workloads.

Prerequisites

Using the PyTorch NGC Container requires the host system to have the following installed:

- [Docker Engine](#)
- [NVIDIA GPU Drivers](#)
- [NVIDIA Container Toolkit](#)

For supported versions, see the [Framework Containers Support Matrix](#) and the [NVIDIA Container Toolkit Documentation](#).

No other installation, compilation, or dependency management is required. *It is not necessary to install the NVIDIA CUDA Toolkit.*

The PyTorch NGC Container is optimized to run on NVIDIA DGX Foundry and NVIDIA DGX SuperPOD managed by NVIDIA Base Command Platform. Please refer to the [Base Command Platform User Guide](#) to learn more about running workloads on BCP clusters.

Running PyTorch Using Docker

To run a container, issue the appropriate command as explained in the [Running A Container](#) chapter in the *NVIDIA Containers For Deep Learning Frameworks*

Catalog > Containers > TensorFlow

TensorFlow

Copy Image Path

Deploy to Vertex AI

Accelerated with

NVIDIA

TensorFlow

Overview

Tags

Layers

Security Scanning

Related Collections

TensorFlow

TensorFlow is an open source platform for machine learning. It provides comprehensive tools and libraries in a flexible architecture allowing easy deployment across a variety of platforms and devices. NGC Containers are the easiest way to get started with TensorFlow. The TensorFlow NGC Container comes with all dependencies included, providing an easy place to start developing common applications, such as conversational AI, natural language processing (NLP), recommenders, and computer vision.

The TensorFlow NGC Container is optimized for GPU acceleration, and contains a validated set of libraries that enable and optimize GPU performance. This container may also contain modifications to the TensorFlow source code in order to maximize performance and compatibility. This container also contains software for accelerating ETL ([DALL·R·APIQS](#)), Training ([cuNN](#), [NCCL](#)), and Inference ([TensorRT](#)) workloads.

Prerequisites

Using the TensorFlow NGC Container requires the host system to have the following installed:

- Docker Engine
- NVIDIA GPU Drivers
- NVIDIA Container Toolkit

For supported versions, see the [Framework Containers Support Matrix](#) and the [NVIDIA Container Toolkit Documentation](#).

No other installation, compilation, or dependency management is required. *It is not necessary to install the NVIDIA CUDA Toolkit.*

Running TensorFlow

To run a container, issue the appropriate command as explained in the [Running A Container](#) chapter in the *NVIDIA Containers For Deep Learning Frameworks User's Guide* and specify the registry, repository, and tags. For more information about using NGC, refer to the [NGC Container User](#)

Description

TensorFlow is an open source platform for machine learning. It provides comprehensive tools and libraries in a flexible architecture allowing easy deployment across a variety of platforms and devices.

Publisher

Google Brain Team

Latest Tag

22.11-tf2-py3

Modified

December 1, 2022

Compressed Size

6.83 GB

Multimedia Support

比较便捷的是，在进行 `docker pull` 的时候，NGC 同时提供了相应的依赖安装。

What Is In This Container?

For the full list of contents, see the [PyTorch Container Release Notes](#).

This container image contains the complete source of the version of PyTorch in `/opt/pytorch`. It is pre-built and installed in Conda default environment (`/opt/conda/lib/python3.8/site-packages/torch/`) in the container image. Visit pytorch.org to learn more about PyTorch.

The NVIDIA PyTorch Container is optimized for use with NVIDIA GPUs, and contains the following software for GPU acceleration:

- [CUDA](#)
- [cuBLAS](#)
- [NVIDIA cuDNN](#)
- [NVIDIA NCCL](#) (optimized for [NVLink](#))
- [RAPIDS](#)
- [NVIDIA Data Loading Library \(DALI\)](#)
- [TensorRT](#)
- [Torch-TensorRT](#)

The software stack in this container has been validated for compatibility, and does not require any additional installation or compilation from the end user. This container can help accelerate your deep learning workflow from end to end.

What is In This Container?

For the full list of contents, see the [TensorFlow Container Release Notes](#).

This container image contains the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module. There are two versions of the container at each release, containing TensorFlow 1 and TensorFlow 2 respectively. Visit [tensorflow.org](#) to learn more about TensorFlow.

The NVIDIA TensorFlow Container is optimized for use with NVIDIA GPUs, and contains the following software for GPU acceleration:

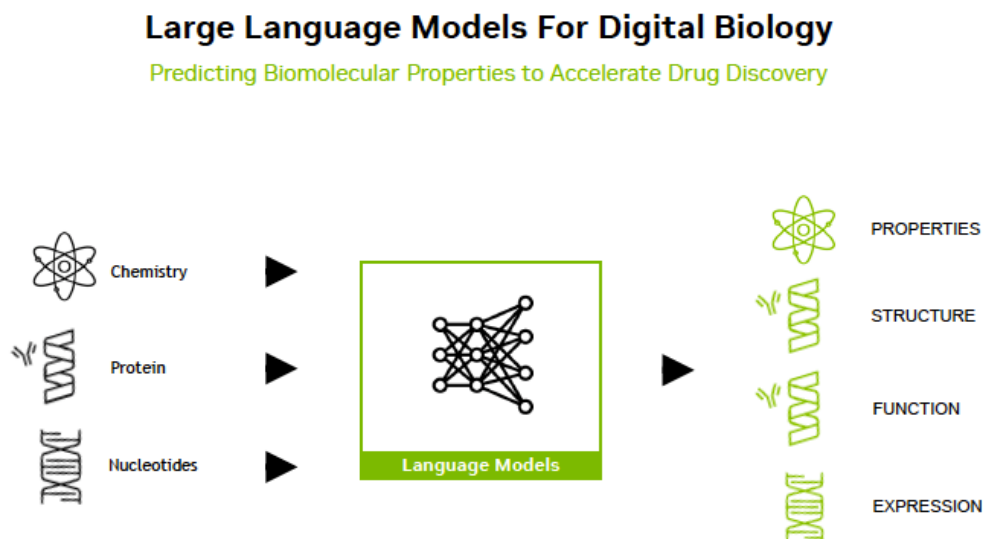
- [CUDA](#)
- [cuBLAS](#)
- [NVIDIA cuDNN](#)
- [NVIDIA NCCL](#) (optimized for [NVLink](#))
- [RAPIDS](#)
- [NVIDIA Data Loading Library \(DALI\)](#)
- [TensorRT](#)
- [TensorFlow with TensorRT \(TF-TRT\)](#)

The software stack in this container has been validated for compatibility, and does not require any additional installation or compilation from the end user. This container can help accelerate your deep learning workflow from end to end.

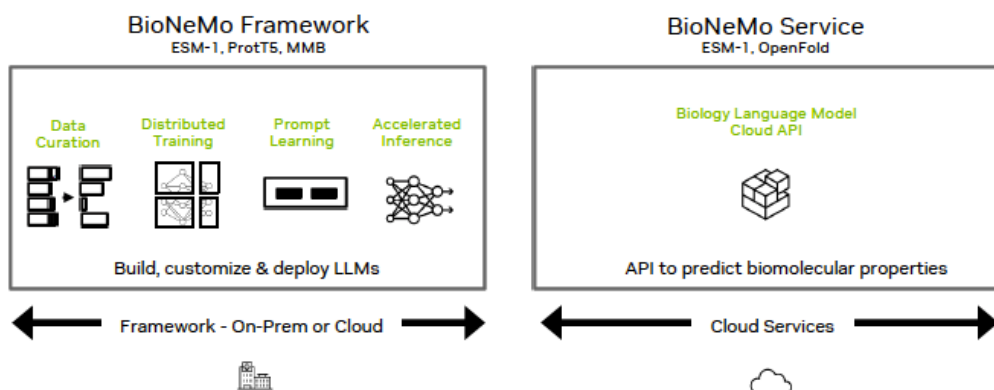
四、BioNeMo

1、什么是 BioNeMo

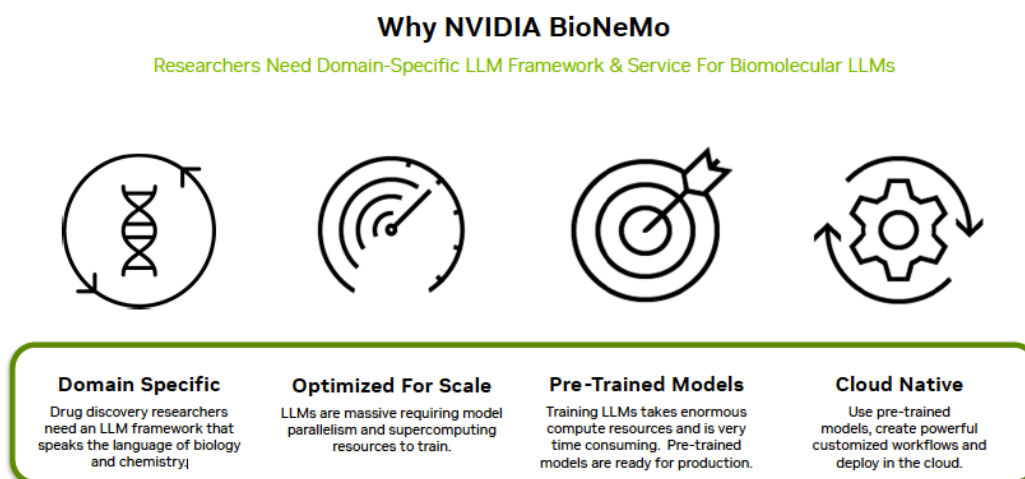
简单来说，BioNeMo 是一个大规模语言模型在生科领域的应用。和最近比较火的文字语言模型 ChatGPT3 不同的是，无论是化学分子式、蛋白质、还是核酸，它实际上也是类似于某种语言：比如化学实际上就是一系列的分子式组成的语言；对于蛋白质来说就是一系列的核苷酸组成的语言；对于核酸来说就是一系列的碱基按照一定的逻辑来组成的语言，其本质上跟文字语言模型有很多相似性，所以把这种大语言模型应用到生科领域，就有了 BioNeMo。



BioNeMo 是生物分子大规模语言模型的框架和云计算 API。不同行业可以通过 BioNeMo 提供的框架，去训练自己行业的模型；BioNeMo 也可以提供服务的方式，供用户在云端进行访问很多预训练的模型。



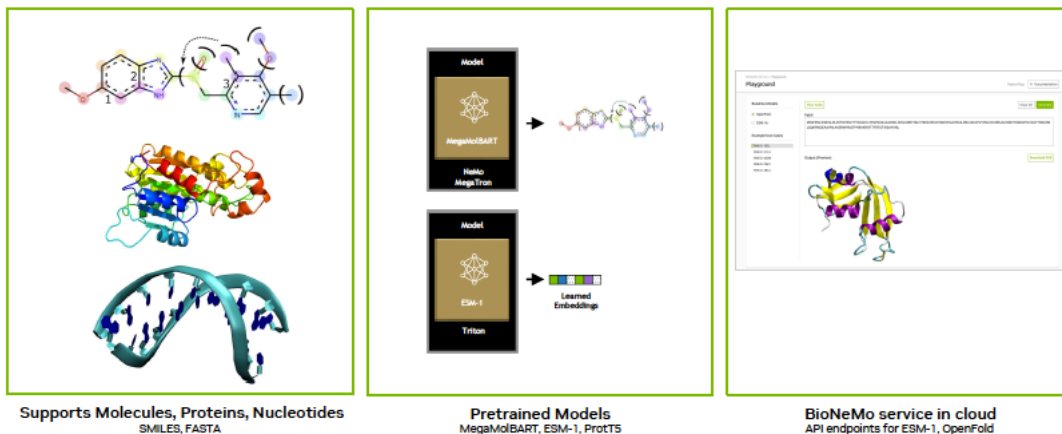
2、为什么要使用 BioNeMo



- 特定领域：药物研发人员需要一个生物化学领域的 LLM 框架；
- 规模优化：针对 LLMs，英伟达有很多并行的优化，无论是训练还是做推理；
- 预训练模型：LLMs 大语言模型的训练过程是一个非常计算密集的过程，BioNeMo 提供了很多预训练模型方便用户试用；
- 云原生：使用预训练模型，用户可以创建强大的自定义工作流并部署在云端。

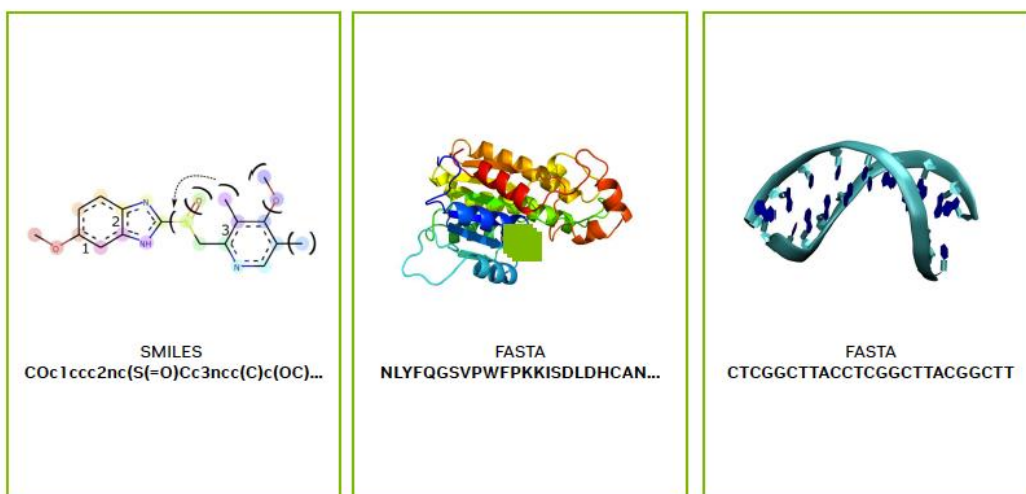
3、BioNeMo 的特征

Top 3 Features of BioNeMo



- 支持分子、蛋白质、核苷酸；
- 提供预训练模型；
- 提供云端服务，如通过简单输入蛋白质氨基酸序列，即可生成合理的蛋白质结构。

Feature 1: Native Support For Molecules, Proteins, And Nucleotides Formats

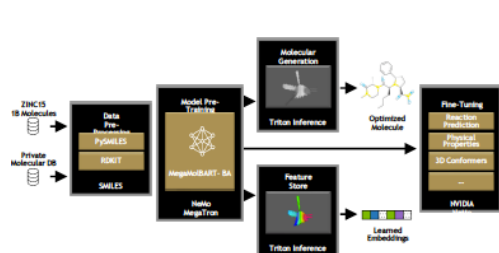


NVIDIA CONFIDENTIAL. DO NOT DISTRIBUTE.

NVIDIA

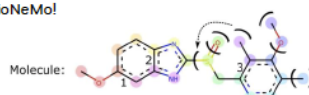
BioNeMo 支持分子、蛋白质、核苷酸

Feature 2: Pre-Trained Models in One Place



ESM-1nv, ProtT5-nv, MegaMolBART

- ESM-1nv is a protein language model for representation learning
- ProtT5-nv is a protein language model for protein generation and representation
- MegaMolBART is a large-scale pre-trained transformer AI model for generative chemistry and representation
- GA in BioNeMo!



SMILES: COc1ccc2nc(S(=O)(=O)Cc3ccc(C)c(OC)c3C)[nH]c2c1



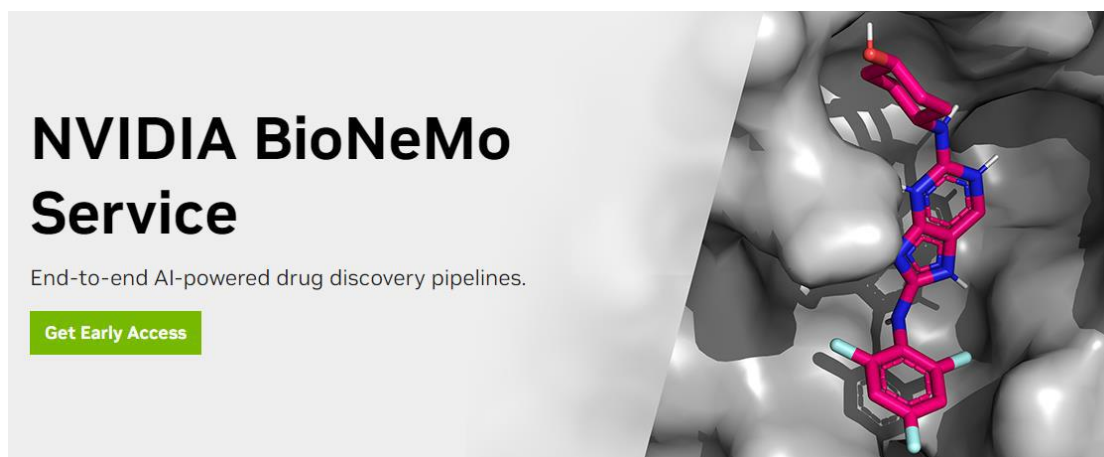
BioNeMo 提供预训练模型

Feature 3: Service With Cloud API



BioNeMo 提供云端服务

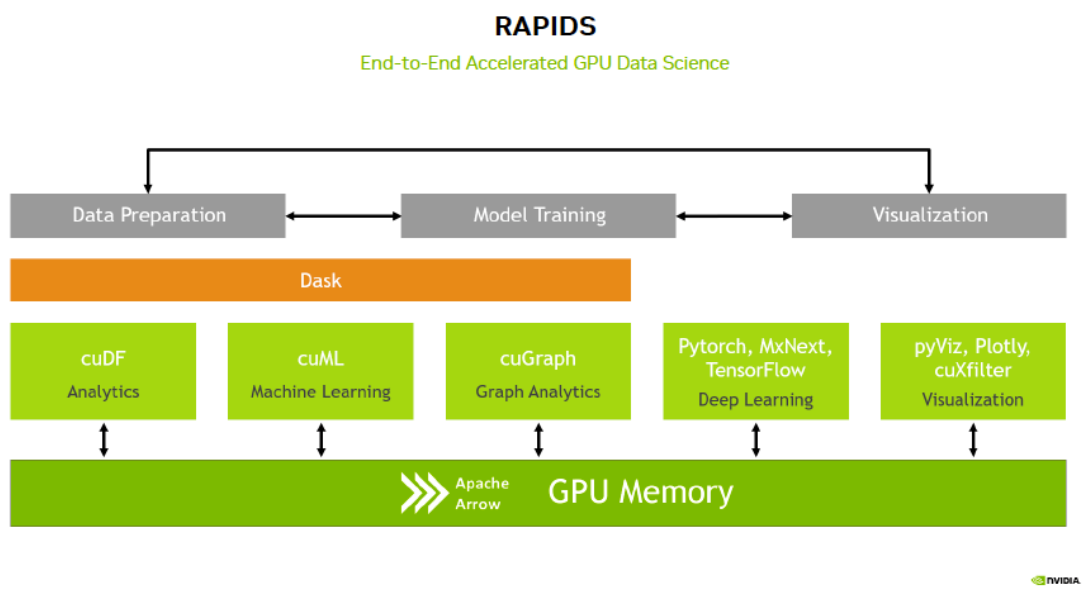
4、申请试用 BioNeMo



申请地址: <https://www.nvidia.com/en-us/gpu-cloud/bionemo/>

五、RAPIDS

1、什么是 RAPIDS



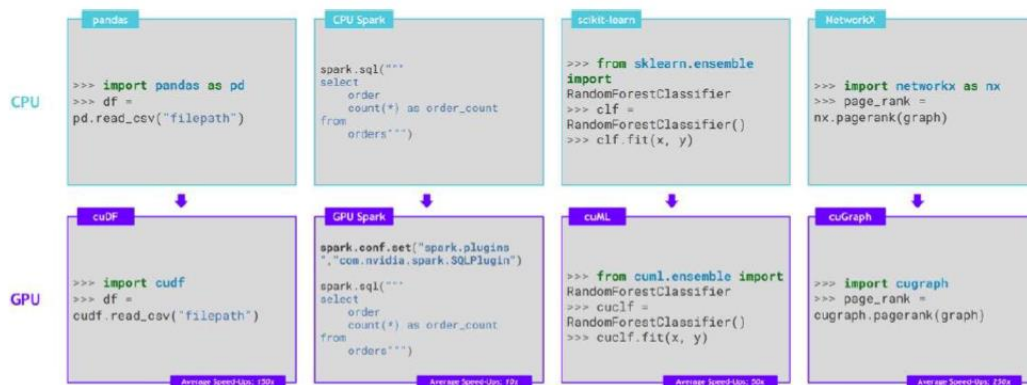
RAPIDS 是帮助在数据科学领域做 GPU 加速的一系列工具的集合,涉及数据准备,模型训练,以及可视化,每一部分都有相应的工具进行支持——

- 数据准备过程中的 cuDF,可以帮助用户取代 DataFrame,接口基本一样,它提供了更好的并行性,整体的访问效率非常高;
- 模型训练部分的 cuML,对标 sklearn,接口也基本一样;
- 其他像图领域的 cuGraph,去做可视化的 cuXfilter 等。

2、CPU 和 GPU (RAPIDS) 方式比较

MINOR CODE CHANGES FOR MAJOR BENEFITS

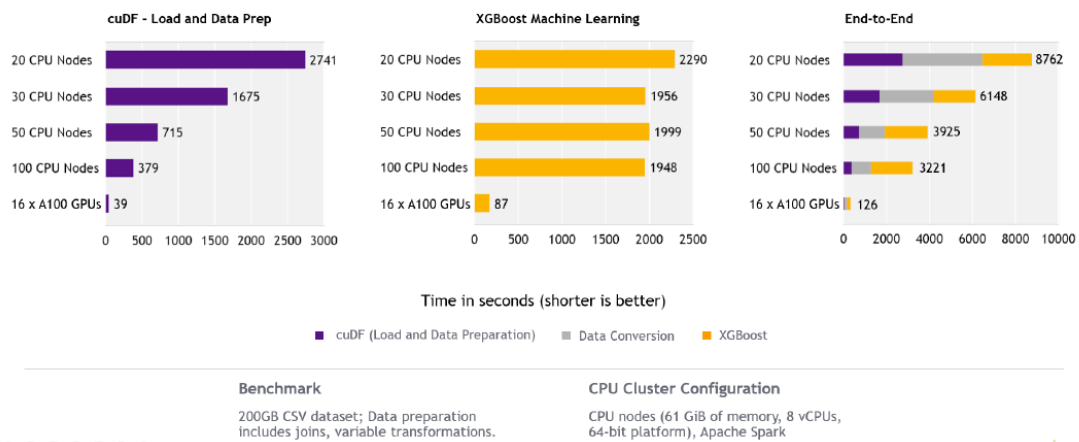
Abstracting Accelerated Compute through Familiar Interfaces



代码对比

通过代码比较可以发现，CPU 的方式和 GPU 的方式基本上是一样的，但 GPU 方式的加速比可以达到几十、上百倍的加速。

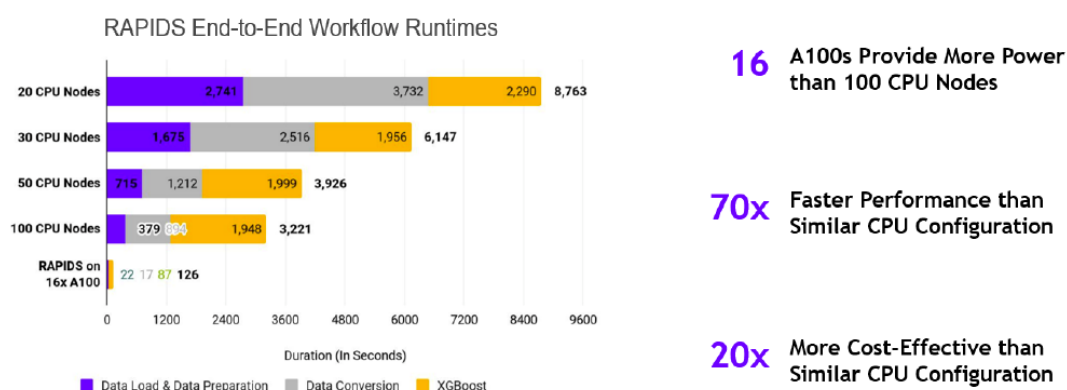
Benchmarks



Benchmark 速度提升对比图

LIGHTNING-FAST END-TO-END PERFORMANCE

Reducing Data Science Processes from Hours to Seconds



端到端性能提升 70 倍，成本下降 20 倍

3、RAPIDS 的使用

RAPIDS Accelerated with NVIDIA

Copy Image Path Deploy to Vertex AI

Overview Tags Layers Security Scanning Related Collections

RAPIDS - Open GPU Data Science

What is RAPIDS?

Visit rapids.ai for more information.

The RAPIDS suite of software libraries gives you the freedom to execute end-to-end data science and analytics pipelines entirely on GPUs. It relies on NVIDIA® CUDA® primitives for low-level compute optimization, but exposes GPU parallelism and high-bandwidth memory speed through user-friendly Python interfaces.

NOTE: Review our prerequisites section below to ensure your system meets the minimum requirements for RAPIDS.

CURRENT VERSION - RAPIDS V22.10

Versions of libraries included in the 22.10 images:

- cuDF [v22.10](#), cuML [v22.10](#), cuGraph [v22.10](#), RMM [v22.10](#), RAFT [v22.10](#), cuSpatial [v22.10](#), cuSignal [v22.10](#), cuXfilter [v22.10](#), c1x [v22.10](#)

IMAGE TYPES

The RAPIDS images are based on [nvidia/cuda](#), and are intended to be drop-in replacements for the corresponding CUDA images in order to make it easy to add RAPIDS libraries while maintaining support for existing CUDA applications.

RAPIDS images come in three types, distributed in two different repos:

This repo ([rapidsai-c1x](#)), contains the following:

同样，可以通过 docker pull 将 RAPIDS 拉取到本地使用。

拓展链接：在 GPU 实例上使用 RAPIDS 加速机器学习任务：

https://help.aliyun.com/document_detail/163838.html

4、RAPIDS 示例

以单细胞基因组分析为例，可以在 NGC 搜索 Single Cell Examples 获取。

Single Cell Examples

Copy Image Path

Deploy to Vertex AI



Description

Contains example notebooks demonstrating the use of RAPIDS for GPU-accelerated analysis of single-cell sequencing data.

Publisher

NVIDIA

Latest Tag

latest

Modified

December 1, 2022

Compressed Size

7.86 GB

Multinode Support

No

Overview

Tags

Layers

Security Scanning

Related Collections

RAPIDS is a suite of open-source Python libraries that can speed up data science workflows using GPU acceleration. Starting from a single-cell count matrix, RAPIDS libraries can be used to perform data processing, dimensionality reduction, clustering, visualization, and comparison of cell clusters.

Several of our examples are inspired by the [Scanpy tutorials](#) and based upon the [AnnData](#) format. Currently, we provide examples for scRNA-seq and scATAC-seq, and we have scaled up to 1 million cells. We also show how to create GPU-powered interactive, in-browser visualizations to explore single-cell datasets.

Dataset sizes for single-cell genomics studies are increasing, presently reaching millions of cells. With RAPIDS, it becomes easy to analyze large datasets interactively and in real time, enabling faster scientific discoveries.

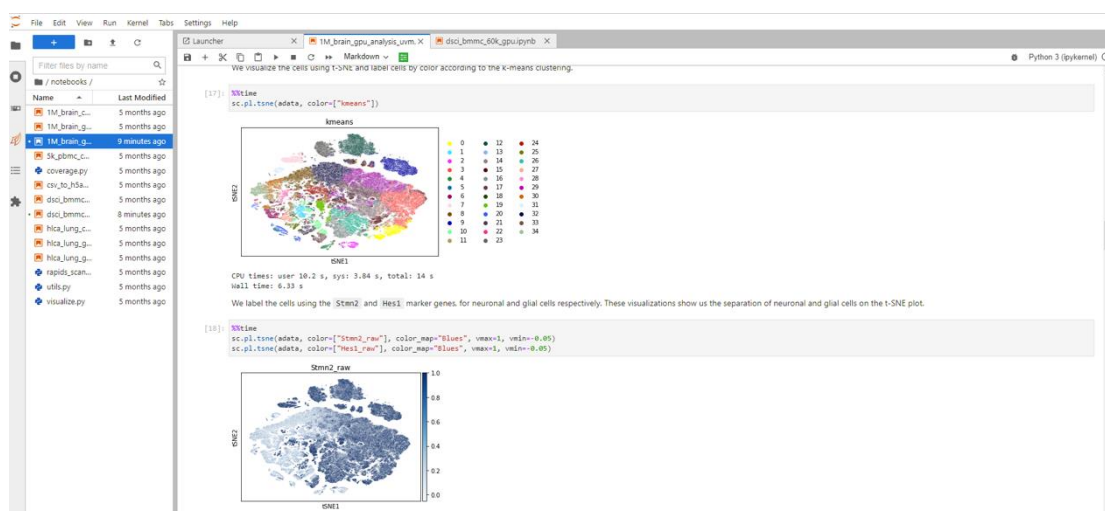
Software Requirements

- Linux OS
- Pascal, Volta, Turing, or an NVIDIA Ampere architecture-based GPU.
- Nvidia Driver
- CUDA 11.4
- Docker
- [Nvidia Docker container toolkit](#)

Hardware Requirements

The sample notebooks are tested using A100 and V100 GPUs. Required number of GPUs depends on the dataset size. The largest dataset referred in the notebooks requires 32GB GPU memory.

- Two V100 or a A100.



启动该示例，使用 Jupyter notebook 远程访问

RAPIDS 加速单细胞基因组分析示例流程图如下：

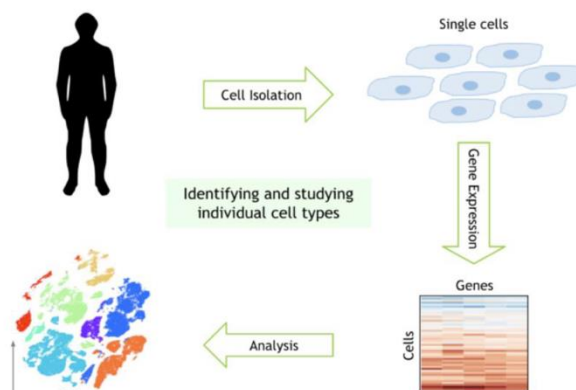


图 1

图 1：展示了单细胞 RNA 测序实验工作流程，首先分离出单个细胞并测各细胞基因活性，具相似基因活性的细胞聚集在一起，以识别群体中各种类型的细胞。

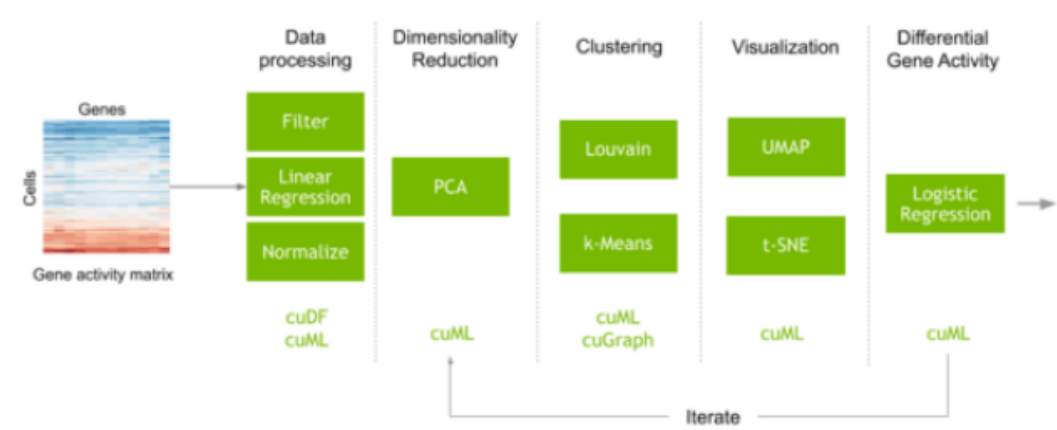


图 2

图 2：展示了单细胞 RNA 测序数据分析步骤，从每个细胞的基因活性矩阵开始，然后 cuDF、cuML，用于进行数据处理，然后降维、聚类、可视化，在不同的族之间发现不同活性的基因差异。

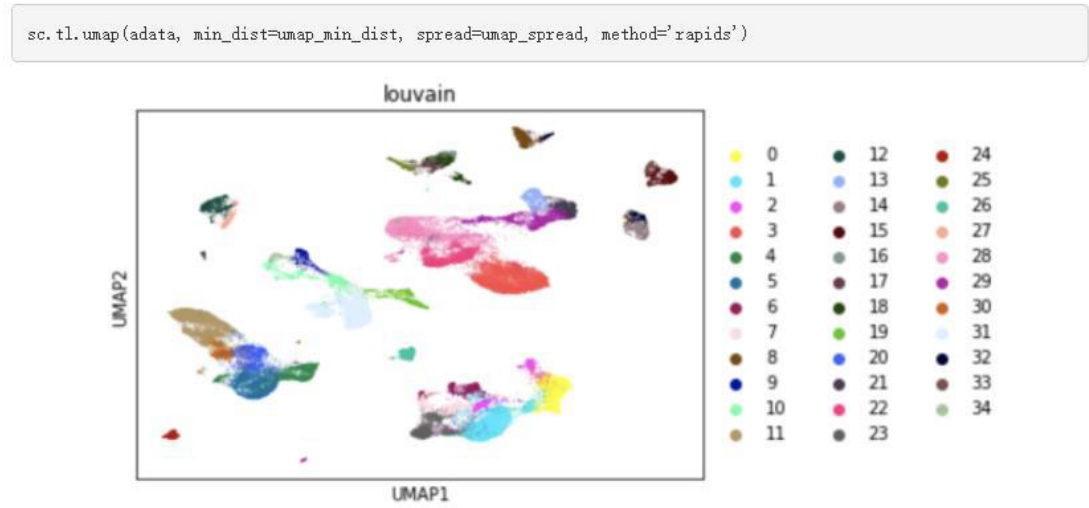


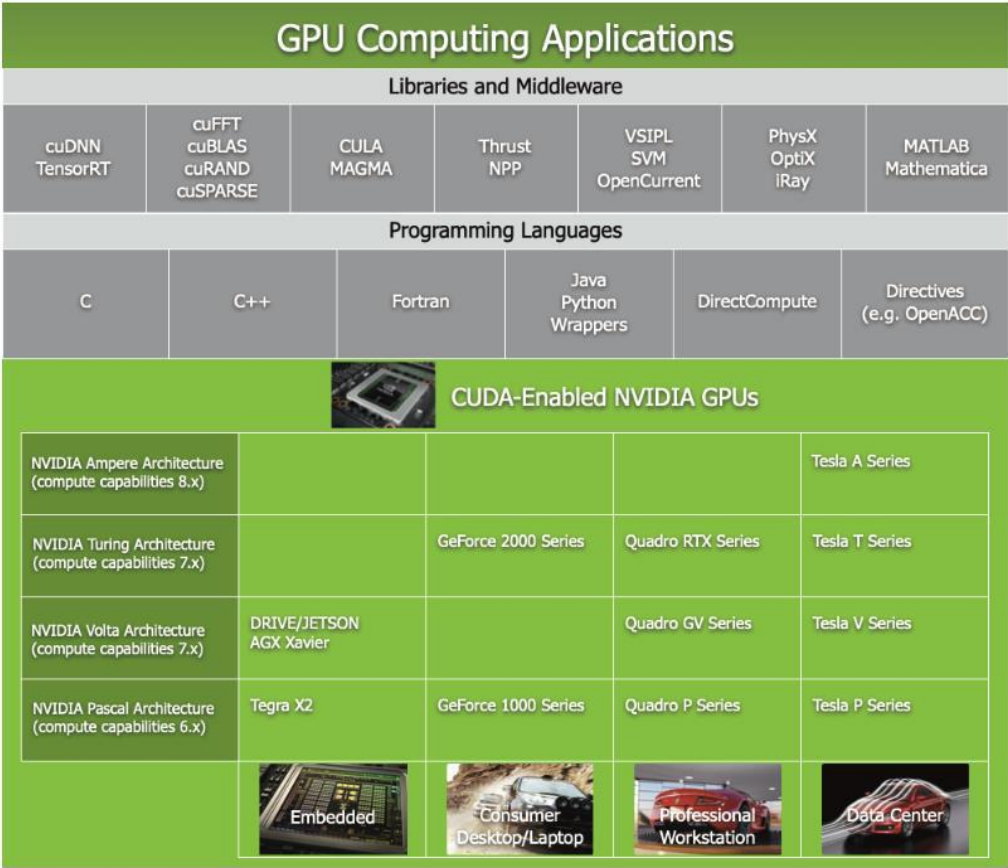
图 3

图 3：用 umap，使用 RAPIDS 可视化，对 7 万个细胞用上述指令进行可视化，只需要 1 秒钟就可以得到这张图；而假如用 CPU 的命令去做可视化的话，则需要差不多 80 秒。

该单细胞基因组分析示例，除了可以在 NGC 获取，也可以通过下面的 github 网址获取：
<https://github.com/NVIDIA-Genomics-Research/rapids-single-cell-examples>

五、CUDA

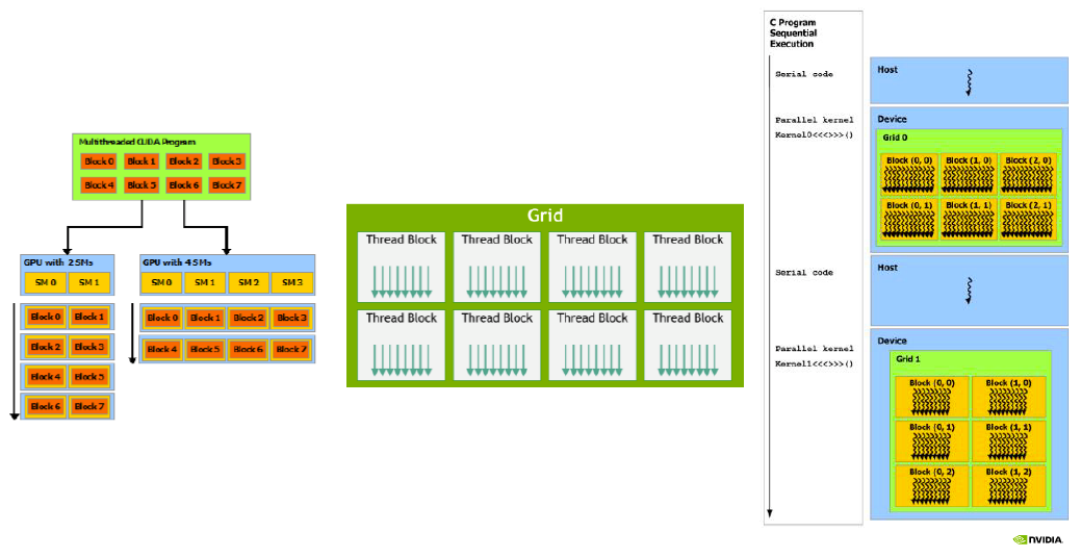
对于 GPU 编程来说，CUDA 是最重要的部分，所有做 GPU 加速的 container，最终反映到的可能都是一些 CUDA，或者 CUDA 相关的一些库。



CUDA 结构

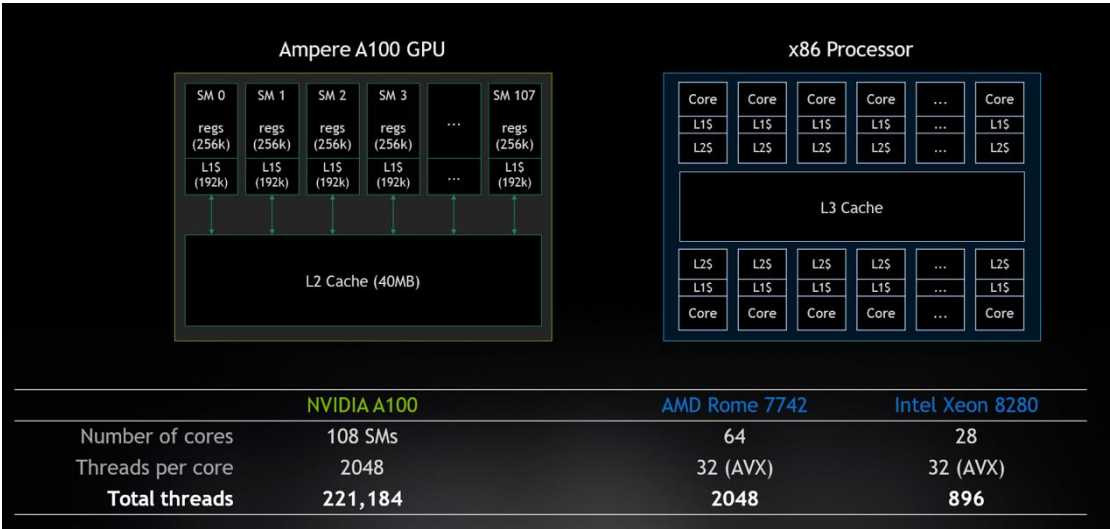
在上面 CUDA 结构中，底层是基于各种各样的显卡即 GPU，CUDA 实际上是和硬件绑定比较紧密的语言，只有熟悉硬件的组成，才能很好的编写 CUDA，可以说 CUDA 对于各种 GPU 加速，它都是一个最基本的支持。

CUDA



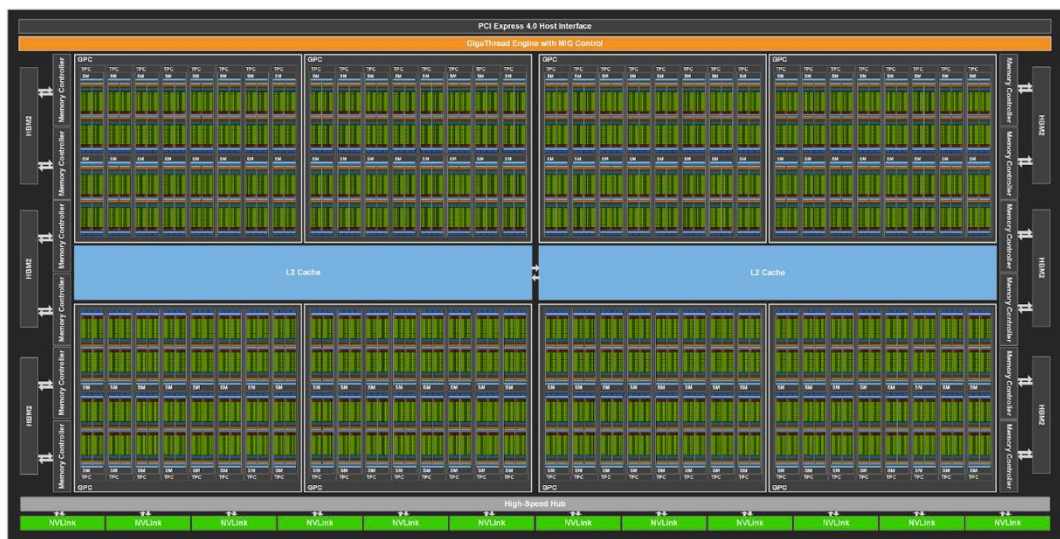
CUDA 的并行执行过程

上图是一条 CUDA 命令的执行过程，它会并行分配给很多的核进行计算，如果程序的本身的并行性非常好，则比较适合用 CUDA 去加速，相当于把程序放到每一个流处理器中进行执行。



GPU 和 CPU 线程对比

上图对 GPU 和 CPU 加速进行对比，对于左侧 A100 GPU 来说，它会有 108 个 SM（流多处理器），每个 SM 里面又可以并行进行 2048 个线程的执行，所以一个 A100 的大约有 22 万个线程，对比右侧 CPU 的 2048 个线程，会有接近 100 倍的差异。所以对于并行性非常好的程序，用 GPU 去进行加速在线程上就会有很大的优势。



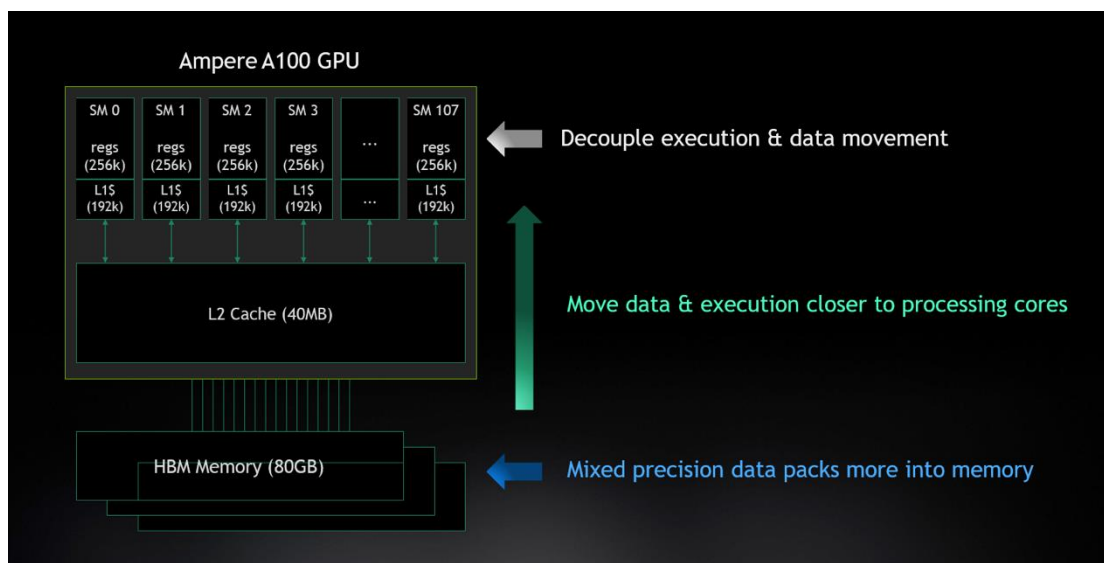
安培 GPU 架构图

在上图安培 GPU 架构图中，左上角的模块就是一个 SM（流多处理器）。



SM（流多处理器）

在每一个 SM 里面都有 64 个 CUDA CORE，用来支持并行运算。



要写好一个 CUDA 程序,就要利用好 GPU 的优势,比如说 GPU 的显存带宽非常大,当然也要注意配合 GPU 自身的算力。另外要注意:解耦执行和数据移动;数据移动和执行放在更接近核心的位置如 L2 Cache 甚至是 L1;将更多的数据打包到内存中等。

扫码了解 NVIDIA 开发者计划和深度学习培训中心 GPU 课程

加入官方群聊, 获取实时更新的 NVIDIA 官方技术培训内容



扫码加入 NVIDIA 开发者计划

您可以访问免费 SDK、技术文档、获取专家帮助和解决重大挑战所需正确硬件的资料



扫码了解 AI 和加速计算实战培训课程

学习 NVIDIA 深度学习培训中心 (DLI) 全球同步课程, 提升开发技能和经验, 获取培训证书