MolAICal 脚本介绍、Linux 版 MolAICal 容器中的程序安装，以及在 Windows 和 macOS 上运行完整版 Linux MolAICal 的教程

Qifeng Bai
邮箱：molaical@yeah.net
主页：https://molaical.github.io 或 https://molaical.gitlab.io

目录

1. 引言

MolAICal 可以根据其命令行逐行执行程序，这有助于 AI 智能体的开发和高效运行。同时，Linux 版本的 MolAICal 将以容器形式运行，使其更具通用性，不再依赖于本地库文件。在 Windows 系统上，用户可以在 Windows 10 或更高版本中通过 Windows 子系统 for Linux（WSL）直接运行 MolAICal，也可以通过 Docker Desktop for Windows - x86_64 部署其镜像来运行 MolAICal（安装指南见：https://docs.docker.com/desktop/setup/install/windows-install/）；或使用 Docker Desktop on Mac (https://docs.docker.com/desktop/setup/install/mac-install)在苹果系统上按照。本文将讨论一些专题内容，例如 MolAICal 批处理脚本和容器安装程序。

---

2. 材料

2.1 软件要求

1) **MolAICal**：https://molaical.github.io 或 https://molaical.gitlab.io

2) **NAMD（CPU 版本）**：https://www.ks.uiuc.edu/Research/namd/
> **注意**：本教程可使用 NAMD 2.x、3.x 或更高版本。例如，如果您使用的是 NAMD 3.x 版本，则教程中的命令 `namd2` 应替换为 `namd3`。对于更高版本的 NAMD，用户可采用类似方式进行替换。

3) **VMD**：https://www.ks.uiuc.edu/Research/vmd

2.2 示例文件

1) 所有必要的教程文件均可从以下地址下载：
https://gitee.com/molaical/tutorials/tree/master/030-MolAICal-special_topics

---

3. 解决 Linux 版本 MolAICal 的库文件依赖问题

Docker 等容器技术具有一个关键优势：它们可以通过挂载方式与宿主机共享文件夹、文件和库文件，从而最大限度地解决诸如库文件依赖等问题。对于部署在容器中的 Linux 版本 MolAICal，若需调用 VMD、NAMD 或 UCSF Chimera 等外部程序，有两种方法：

1) **直接在 MolAICal 容器内安装外部程序（如 VMD、NAMD、UCSF Chimera）并配置内部文件**（推荐，见 3.1 节）

2) **通过挂载方式共享宿主机上的外部软件及其依赖库文件**（见 3.2 节）

3.1 在 MolAICal 容器内安装外部程序（推荐）

a) **首先，将文件复制到 MolAICal 容器中**

```
# 进入容器文件系统（将进入 "/root" 目录）
#> molaical.exe -eset shell in

# 将 VMD 和 NAMD 安装包从本地机器复制到容器中，'cp' 命令的第一部分（源路径）
# 位于本地主机，第二部分（目标路径）在容器内；VMD 和 NAMD 软件包可通过以下命
# 令移入容器。
#> cp /home/user/<本地文件> /root/soft

# 退出容器文件系统
#> exit
```

2. **其次，进入 MolAICal 容器的虚拟环境**

```
#> molaical.exe -eset sys run molaical
```

注：molaical 是容器名称。

3. **在 MolAICal 容器虚拟环境中安装软件的方式与在本地主机上安装相同。以下以安装 VMD 和 NAMD 为例：**

1) 安装 NAMD：

解压 NAMD 文件（假设解压后的文件夹名为 namdcpu），然后使用以下命令将其路径告知 MolAICal：

```
#> molaical.exe -call set -n NAMD -p "/root/soft/namdcpu/namd3"
```

注：请将上述 VMD 和 NAMD 的路径替换为您系统中的实际路径。-n 后面的 "VMD" 和 "NAMD"（大小写不敏感）是固定的标识符。为确保 MM/GBSA 结果的可重复性，建议使用 NAMD 的 CPU 版本，因为 CUDA 版本中的 seed 参数似乎对结果可重复性无效。

2) 安装 VMD：

按以下步骤操作：

◆ **解压 VMD 文件：**

```
#> tar -xzvf vmd-xxx.tar.gz
```

注：请将上述路径替换为您系统中的实际路径。

◆ **修改 VMD 解压目录中名为 configure 的文件中的安装路径：**

```
# 默认值：
$install_bin_dir="/usr/local/bin";
$install_library_dir="/usr/local/lib/$install_name";

# 修改为：
$install_bin_dir="/root/soft/vmd193/bin";
$install_library_dir="/root/soft/vmd193/lib/$install_name";
```

◆ **安装 VMD：**

```
#> cd vmd-xxx
#> ./configure LINUXAMD64
#> cd src
#> make install
```

注：请运行 ./configure 并根据所用计算机选择正确的类型，此处为 "LINUXAMD64"。

◆ **然后使用以下命令将 VMD 路径告知 MolAICal：**

```
#> molaical.exe -call set -n VMD -p "/root/soft/vmd193/bin/vmd"
```

至此，NAMD 和 VMD 在 MolAICal 容器内的安装与配置已完成。

✧ **记得使用 exit 命令退出 MolAICal 虚拟环境，返回本地计算机进行计算（主要是为了省去文件拷贝步骤；在容器内运行也可行，但需手动将数据从本地计算机复制到 MolAICal 容器中）。**

```
#> exit
```

3) 保存并加载已修改的容器（可选）

为保留容器内所做的更改，并避免日后重新安装软件（因为一旦容器被删除，所有数据都会丢失），可执行以下操作：

✧ **获取容器 ID 和名称：**

```
#> molaical.exe -eset sys ps
```

✧ **克隆该容器：**

```
#> molaical.exe -eset sys export --clone -o molaicalv2.tar molaical
```

注：molaical 是容器名称，也可使用容器 ID。如果报错，那可能是因为挂载的文件系统的权限问题，可以忽略。

✧ **假如需要，导入克隆的容器：**

```
#> molaical.exe -eset sys import --clone --name molaicalv2 molaicalv2.tar
```

✧ **将新导入的容器名称更改为默认容器名 "molaical"，原容器重命名为 "bakmolaical"：**

```
#> molaical.exe -eset sys rename molaical bakmolaical
#> molaical.exe -eset sys rename molaicalv2 molaical
```

注意：容器（动态）是从镜像（静态）生成的。软件安装等操作必须在动态容器中进行；如果克隆了该容器，恢复时仍基于原始底层镜像。

更多详情请参阅 MolAICal 用户手册，或运行以下命令获取帮助：

```
#> molaical.exe -eset sys --help
```

3.2 共享宿主机上的外部软件及其依赖库文件

如果本地计算机已安装 VMD、NAMD 等软件，且用户不想在 MolAICal 容器虚拟环境中再次安装，可通过挂载方式将本地相关必要文件、库等共享给 MolAICal 容器。

以下以 Ubuntu 系统为例。对于其他系统，请先定位对应库文件的位置——请注意，不同系统的库文件位置可能不同。如果之前的路径混乱，可使用以下命令清除：

```
#> molaical.exe -eset clean
#> molaical.exe -call set -o clean
```

用户可通过以下命令查看 MolAICal 的外部路径：

```
#> molaical.exe -call set -o all
```

a) **将 NAMD 和 VMD 的本地路径告知 MolAICal 容器：**

```
#> molaical.exe -call set -n VMD -p "/home/feng/soft/vmd193/bin/vmd"
#> molaical.exe -call set -n NAMD -p "/home/feng/soft/namdcpu/namd3"
```

2. **将本地机器上的必要库文件告知 MolAICal 容器：**

```
#> molaical.exe -eset lib /usr/lib/x86_64-linux-gnu
#> molaical.exe -eset bin /usr/lib/x86_64-linux-gnu
#> molaical.exe -eset mount /usr/lib/x86_64-linux-gnu
#> molaical.exe -eset lib /usr/lib
#> molaical.exe -eset bin /usr/lib
#> molaical.exe -eset mount /usr/lib
```

随后，进入材料文件夹 `"020-mmpbsa"` 并进行测试：

```
#> molaical.exe -mmpbgbsa -f mmpbsa_apbs.vmd
```

---

4. MolAICal 批处理脚本

当 -c 的值为 "script" 时，MolAICal 将运行脚本（在 Windows DOS 或 Linux 命令控制台中运行的命令列表）。

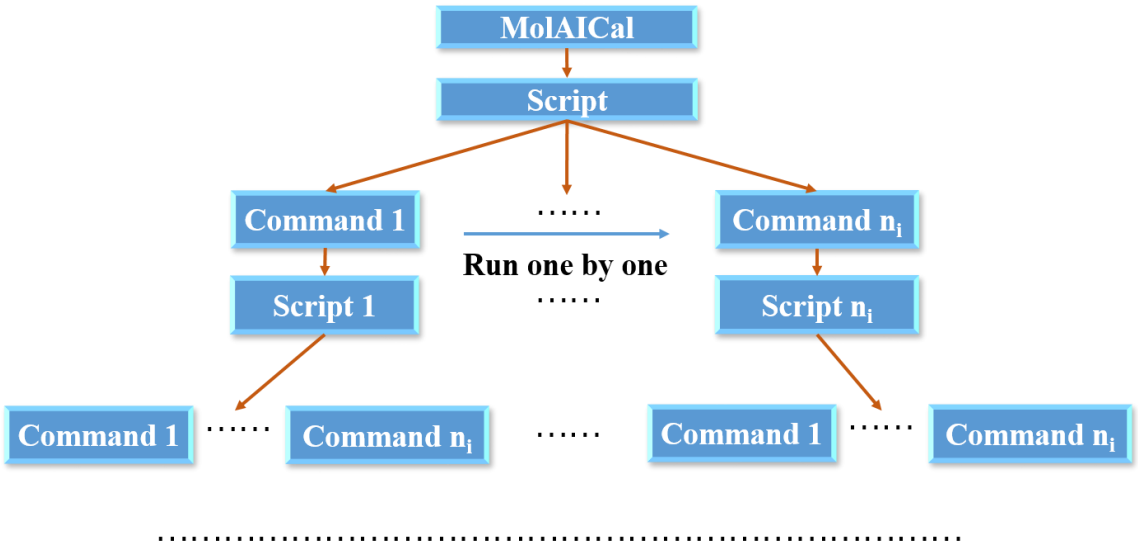

图 4.1.1：MolAICal 脚本框架

4.1 MolAICal 脚本原理

图 4.1.1 展示了 MolAICal 脚本执行过程的结构化视图。以下是详细的专业描述：

a) MolAICal 作为主模块，启动主脚本（Script）的执行。
b) 脚本包含顺序执行的命令，例如 Command 1，它会触发 Script 1 的执行。
   - 首先执行 Command 1，然后执行 Script 1。
   - Script 1 执行完毕后，可能会循环返回以执行更多命令，包括再次执行 Command 1 或其他命令（如 Command n_i）。
c) Command n_i 可以是脚本中的不同操作，也会触发 Script n_i 的执行。
d) 此设置展示了一种递归、模块化的流程，其中命令与脚本相互作用并触发后续操作，支持动态执行流。

蓝色箭头表示命令与其对应脚本之间的直接连接，水平蓝色箭头表示命令执行之间的流程。

4.2 MolAICal 脚本参数说明

-i 后面的字符串为参数列表，将按空格分割。分割后的字符串将依次替换脚本文件中的 $molargs1、$molargs2、$molargs3 等变量。$molargs1、$molargs2、$molargs3 等变量的数量与 -i 后分割出的字符串数量一致。

某些特殊字符的表示方式如下：

a) 若某个 `$molargs[x]`（x = 1, 2, ...）的值为 `"^*^"`，则代表注释符号（例如 `#`）。
b) 若某个 `$molargs[x]` 的值为 `"^^^"`，则代表一个空格字符 `" "`。

示例：

```
# 1. 不修改参数，直接运行 'runscript' 文件
#> molaical.exe -call run -c script -n runscript

# 2. 将 'runscript' 文件中的 $molargs1 替换为 complex.dcd
#> molaical.exe -call run -c script -n runscript -i complex.dcd

# 3. 将 'runscript' 文件中的 $molargs1 和 $molargs2 分别替换为 complex.dcd 和 complex.psf
#> molaical.exe -call run -c script -n runscript -i complex.dcd complex.psf
```

`runscript` 文件内容示例（仅为示例）：

```
# 1. fix autopsf
molaical.exe -call run -c fixpsf

# 2. generate psf
molaical.exe -call run -c vmdargs -i -pdb $molargs1 -dispdev text -args autopsf autopsf -mol 0 -prefix protein
molaical.exe -call run -c vmdargs -i -pdb $molargs2 -dispdev text -args autopsf autopsf -mol 0 -top $molargs3 -prefix ligand

# 3. merge file
molaical.exe -call run -c merge -i -dispdev text -args -first ligand_formatted_autopsf.psf ligand_formatted_autopsf.pdb -second protein_formatted_autopsf.psf protein_formatted_autopsf.pdb -output complex_final

# 4. default to run 20 ps minimization and 40 ps MD simulation
molaical.exe -call run -c runnamd -i -dispdev text -args -s complex_final.psf -c complex_final.pdb -nf md_configure.conf -output_prefix com_md

# 5. Cut last 10 steps for MM/GBSA calculation
molaical.exe -call run -c catdcd -i -dispdev text -args -out result.dcd -first 51 -last 60 -stride 1 com_md.dcd

# 6. Calculate MM/GBSA
molaical.exe -call run -c mmpbgbsa -i -dispdev text -args -s complex_final.psf -c complex_final.pdb -t result.dcd -cs "segname,AP1,CO1" -rs "segname,AP1" -ls "segname,CO1" -pb 2 -ff $molargs3
```

```
# 7. delete file with command (in windows: del xxx, in linux: rm xxx)
rm *md.* *run.* *formatted* result.dcd *final.* *_md* *_wb*
```

如需可运行的完整示例，请前往示例材料 `"021-single_mmpbgbsa"` 目录，运行 MolAICal 脚本案例。

---

5. 在 Windows 和 macOS 操作系统上运行完整版 Linux MolAICal

下载 Docker Desktop:

- 在 Mac 上安装 Docker Desktop：
  https://docs.docker.com/desktop/setup/install/mac-install
- 在 Windows 上安装 Docker Desktop：
  https://docs.docker.com/desktop/setup/install/windows-install

1) 安装 Docker Desktop 非常简单：只需点击下载的可执行文件 Docker Desktop，并按照提示完成安装。

2) 由于 Linux 版本的 MolAICal 目前使用独立的 udocker 镜像，在 Windows 和 macOS 操作系统上安装 Docker Desktop 后，即可运行完整版 Linux MolAICal。在 Windows DOS 或 macOS 终端上运行的通用步骤如下：

a) 拉取 Ubuntu 20.04

```
#> docker pull ubuntu:20.04
```

注意：如果您所在位置无法连接到 Docker 的镜像仓库，会遇到以下错误：

Error response from daemon: Get "https://registry-1.docker.io/v2/": net/http: request canceled while waiting for connection (Client.Timeout exceeded while awaiting headers).

这种情况下，用户需要根据下面的方法更新镜像仓库源（参见本教程中的附录 1）。如果没有问题，请跳过附录 1。

b) 基于此镜像启动一个用于运行 MolAICal 的容器

```
#> docker run -it -d --name molaical -w /root ubuntu:20.04 /bin/bash
#> docker exec -it molaical /bin/bash
#> apt update
```

```
#> apt install make -y
#> apt install vim -y
```

注意：如果需要安装程序，应在终端中输入并运行"apt update"。

c) 将 MolAICal 和实验文件复制到 Docker 中

```
#> docker cp <您的路径>\MolAICal-linux64 molaical:/root/soft
```

至此，MolAICal 和材料已复制到名为"molaicalvx"的容器中的/root/目录。

d) 进入 Docker 环境，安装 MolAICal，并运行 MolAICal 案例

```
***** 进入 Windows 或 MacOS Docker 容器 *****
#> docker exec -it molaical /bin/bash

***** 进入 molaical 容器，如果需要安装外部程序 *****
#> molaical.exe -eset sys run molaical
#> exit
#> mkdir workdir
#> cd /root/workdir

***** 生成肽段的例子 *****
#> molaical.exe -call run -c pepgen -i -l 8 -n 2 -3d -nc -o linear_seq.
txt -dir linear
```

---

6. Linux 版 MolAICal：架构与加速方案

如图 6.1 所示，MolAICal 容器不仅能调用容器系统的文件和程序，还能通过挂载与映射机制访问宿主机库文件，**有效解决了库冲突或依赖缺失等问题**。然而，**容器内程序的执行速度确实可能低于本地环境。**

图 6.1 表明：由于映射关系的存在，容器中"molalcal.exe"执行的计算命令与本地"molaical.exe"运行命令完全一致。

- 因此，对于库依赖较少的任务（如：使用 MolAICal 进行虚拟筛选），为了提升筛选速度，议将 MolAICal 从容器复制到本地：

```
# 1. 进入容器文件系统（默认进入"/root"目录）
#> molaical.exe -eset shell in

# 2. 'cp'命令前半部分指向容器路径，后半部分指向宿主机路径
#> cp -r soft/MolAICal-linux64 <local machine directory>/
```

```
# 3. 退出容器文件系统
#> exit
```

◆ 最终通过相同方法在本地执行虚拟筛选，可显著加速筛选过程。

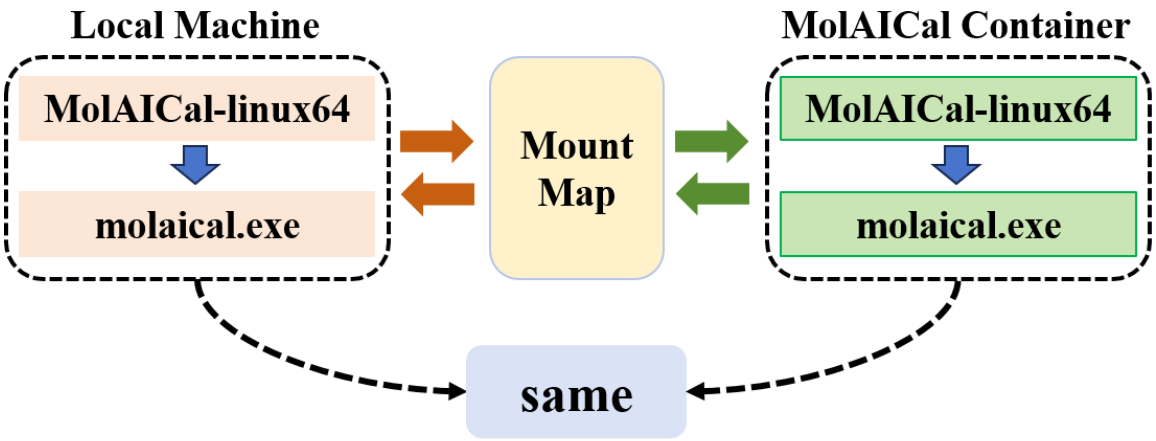

图 6.1 MolAICal Linux 版本架构图

附录 1

通过点击**设置 → Docker Engine** 来配置 Docker Desktop，"添加内容"后点击**应用并重启**，然后在左侧框中输入仓库源（见图 a1）。

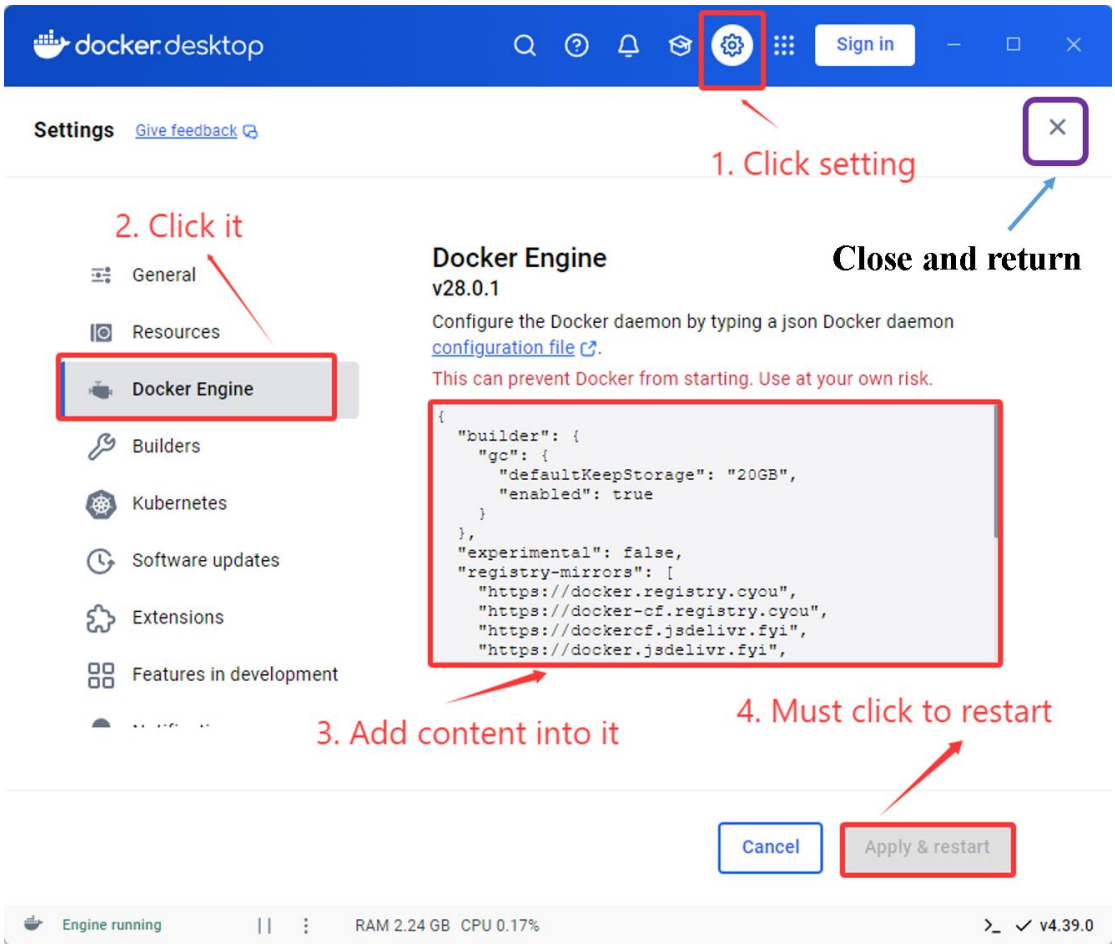

图 a1

镜像仓库源的配置内容通常为：

- 每个项目应该用逗号","分隔。
- "registry-mirrors"字段告诉 Docker Desktop 新的镜像仓库源。此处提供的示例为中国镜像源，在其他地区或国家可能无法保证工作。如果不可用，请按照格式替换为您能够连接的合适仓库源。

输入的仓库源内容如下：

```
{
 "builder": {
```

```
  "gc": {
    "defaultKeepStorage": "20GB",
    "enabled": true
  }
 },
 "experimental": false,
 "registry-mirrors": [
  "https://docker.registry.cyou",
  "https://docker-cf.registry.cyou",
  "https://dockercf.jsdelivr.fyi",
  "https://docker.jsdelivr.fyi",
  "https://dockertest.jsdelivr.fyi",
  "https://mirror.aliyuncs.com",
  "https://dockerproxy.com",
  "https://mirror.baidubce.com",
  "https://docker.m.daocloud.io",
  "https://docker.nju.edu.cn",
  "https://docker.mirrors.sjtug.sjtu.edu.cn",
  "https://docker.mirrors.ustc.edu.cn",
  "https://mirror.iscas.ac.cn",
  "https://docker.rainbond.cc"
 ]
}
```