- Home
- Overview
- Publications
- Research
- **Software**
  - NAMD
  - VMD
  - GPU Computing
  - Lattice Microbes
  - Atomic Resolution Brownian Dynamics
  - MDFF
  - QwikMD
  - Other
- Outreach

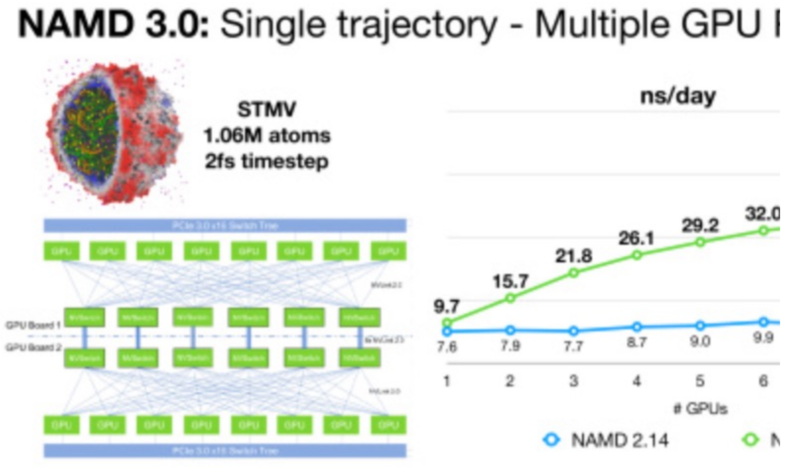
NAMD 3.0 Alpha, GPU-Resident Single-Node-Per-Replicate Test Builds

This page contains special developmental early "alpha" test builds of NAMD 3.0 intended to provide very early testing access to the NAMD user and developer communities.

- **NAMD 3 alpha test downloads**
- **NAMD 3 usage notes**

An exciting NAMD 3.0 feature being developed is a GPU-resident single-node-per-replicate computation mode that can speed up simulations up to two or more times for modern GPU architectures, e.g., Volta, Turing, Ampere.

This new NAMD simulation mode maximizes performance of small- to moderate-sized molecular dynamics simulations suitable for the computational capabilities of a single GPU-accelerated compute node, by virtue of new GPU-specific streamlined code paths that offload the integrator and rigid bond constraints to the GPU, while bypassing and thereby eliminating CPU activities and associated overheads that would otherwise slow down these simulations. With current developmental test builds of NAMD 3.0 alpha, the new code paths can provide a 2x performance gain compared to previous versions of NAMD running the same simulation on the same hardware.



Early multi-GPU-per-replicate scaling plot for ST on an NVIDIA DGX-2 w/ Tesla V100 GPUs.

What MD Simulations Are Well-Suited to NAMD 3.0 Alpha Versions?

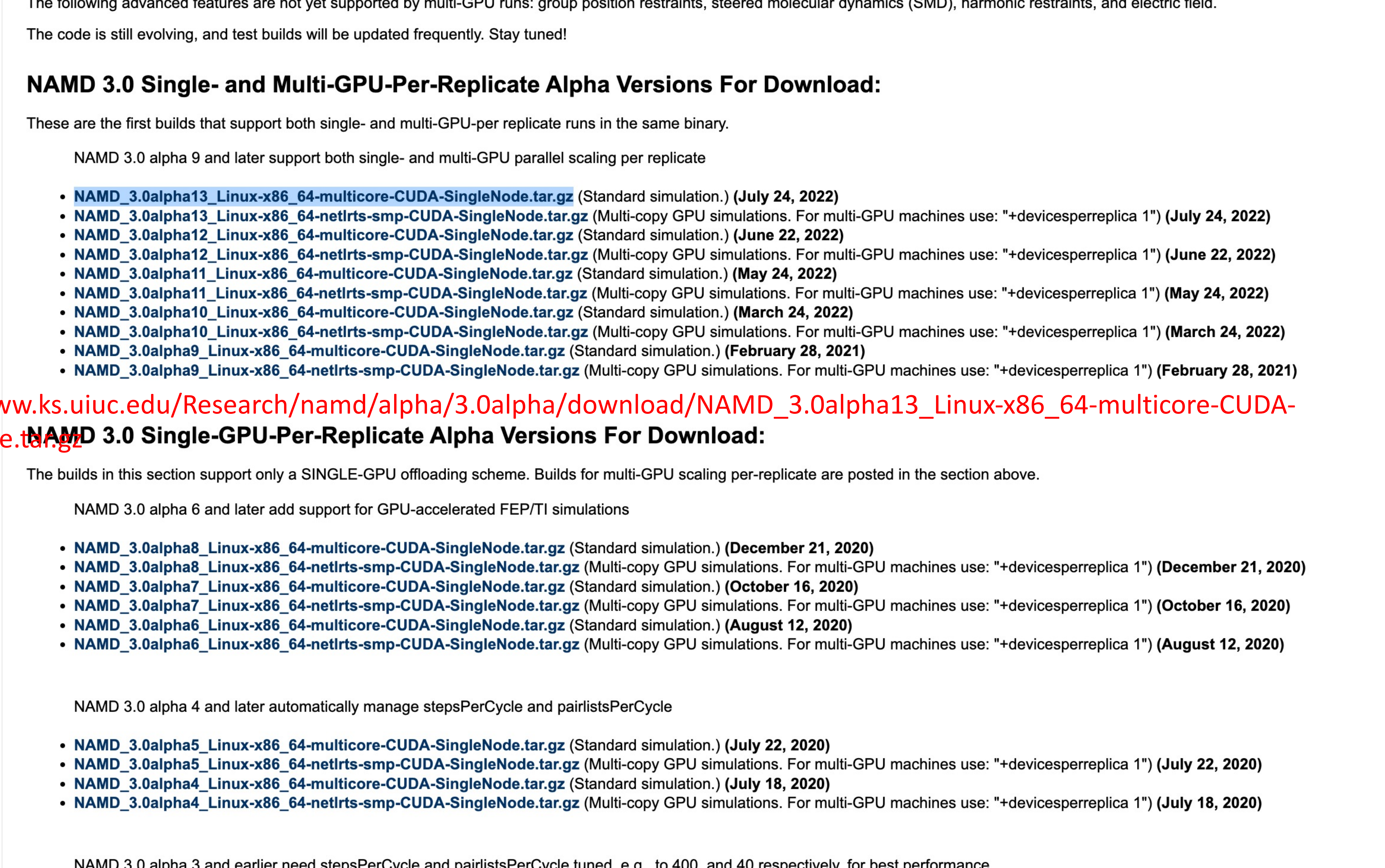
This scheme is intended for small to medium systems (10 thousand to 1 million atoms). For larger simulations, you should stick to the regular integration scheme, e.g., as used in NAMD 2.x.

This scheme is intended for modern GPUs, and it might slow your simulation down if you are not running on a Volta, Turing, or Ampere GPU! If your GPU is older, we recommend that you stick to NAMD 2.x.

The single-node version of NAMD 3.0 has almost everything offloaded to the GPU, so large CPU core counts are NOT necessary to get good performance. We recommend running NAMD with a low +p count, maybe 2-4 depending on system size, especially if the user plans on running multiple replica simulations within a node.

NAMD 3.0 Feature Notes:

NAMD contains a lot of code! We currently don't have corresponding GPU code for everything that the original CPU is capable of handling. The consequence is that some features are still not supported, and others are supported but don't gain much performance (yet). Most of the supported features are related to equilibrium simulations, but some biasing schemes



The following advanced features are not yet supported by multi-GPU runs: group position restraints, steered molecular dynamics (SMD), harmonic restraints, and electric field.

The code is still evolving, and test builds will be updated frequently. Stay tuned!

NAMD 3.0 Single- and Multi-GPU-Per-Replicate Alpha Versions For Download:

These are the first builds that support both single- and multi-GPU-per replicate runs in the same binary.

NAMD 3.0 alpha 9 and later support both single- and multi-GPU parallel scaling per replicate

- **NAMD_3.0alpha13_Linux-x86_64-multicore-CUDA-SingleNode.tar.gz** (Standard simulation.) **(July 24, 2022)**
- **NAMD_3.0alpha13_Linux-x86_64-netlrts-smp-CUDA-SingleNode.tar.gz** (Multi-copy GPU simulations. For multi-GPU machines use: "+devicesperreplica 1") **(July 24, 2022)**
- **NAMD_3.0alpha12_Linux-x86_64-multicore-CUDA-SingleNode.tar.gz** (Standard simulation.) **(June 22, 2022)**
- **NAMD_3.0alpha12_Linux-x86_64-netlrts-smp-CUDA-SingleNode.tar.gz** (Multi-copy GPU simulations. For multi-GPU machines use: "+devicesperreplica 1") **(June 22, 2022)**
- **NAMD_3.0alpha11_Linux-x86_64-multicore-CUDA-SingleNode.tar.gz** (Standard simulation.) **(May 24, 2022)**
- **NAMD_3.0alpha11_Linux-x86_64-netlrts-smp-CUDA-SingleNode.tar.gz** (Multi-copy GPU simulations. For multi-GPU machines use: "+devicesperreplica 1") **(May 24, 2022)**
- **NAMD_3.0alpha10_Linux-x86_64-multicore-CUDA-SingleNode.tar.gz** (Standard simulation.) **(March 24, 2022)**
- **NAMD_3.0alpha10_Linux-x86_64-netlrts-smp-CUDA-SingleNode.tar.gz** (Multi-copy GPU simulations. For multi-GPU machines use: "+devicesperreplica 1") **(March 24, 2022)**
- **NAMD_3.0alpha9_Linux-x86_64-multicore-CUDA-SingleNode.tar.gz** (Standard simulation.) **(February 28, 2021)**
- **NAMD_3.0alpha9_Linux-x86_64-netlrts-smp-CUDA-SingleNode.tar.gz** (Multi-copy GPU simulations. For multi-GPU machines use: "+devicesperreplica 1") **(February 28, 2021)**

https://www.ks.uiuc.edu/Research/namd/alpha/3.0alpha/download/NAMD_3.0alpha13_Linux-x86_64-multicore-CUDA-SingleNode.tar.gz

NAMD 3.0 Single-GPU-Per-Replicate Alpha Versions For Download:

The builds in this section support only a SINGLE-GPU offloading scheme. Builds for multi-GPU scaling per-replicate are posted in the section above.

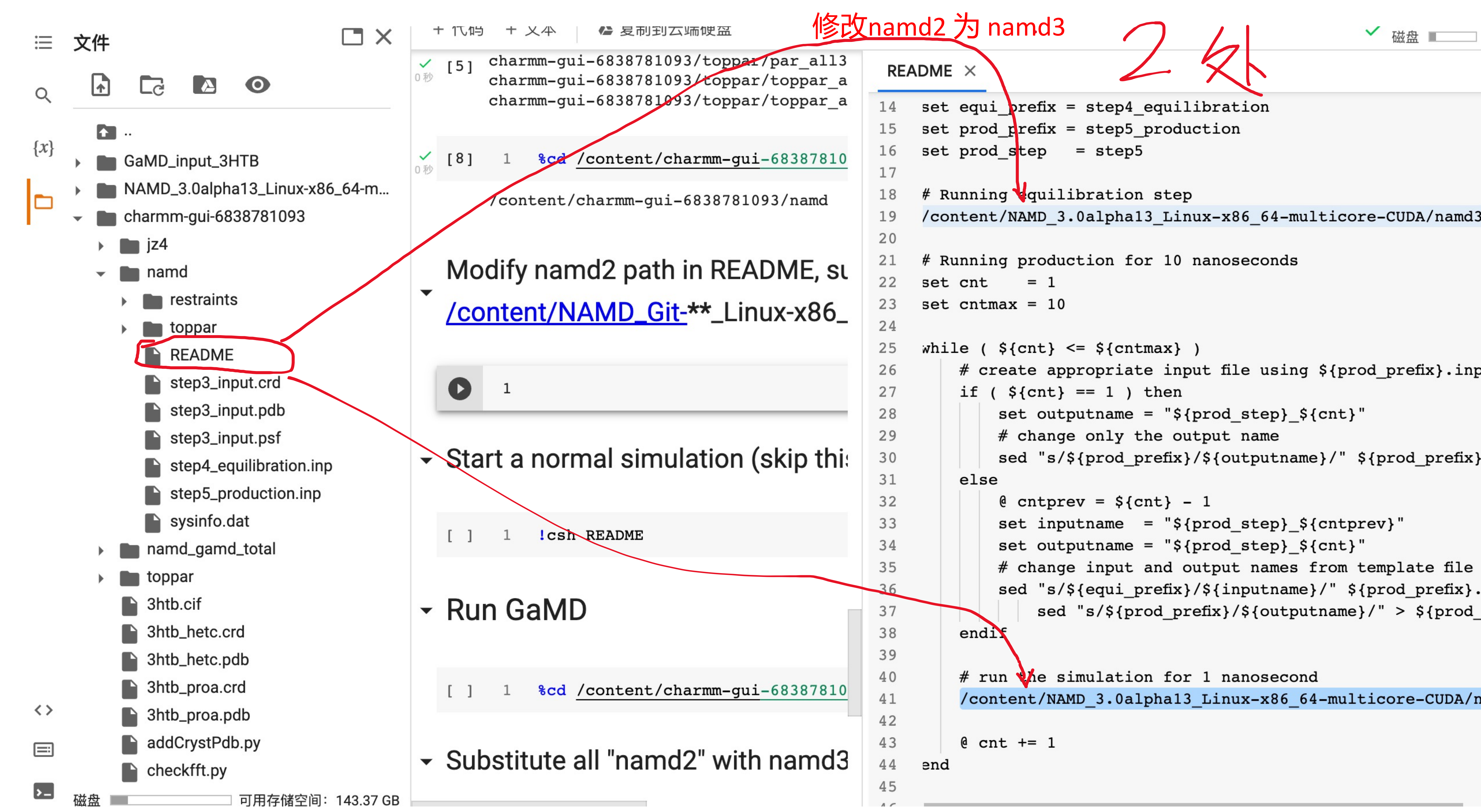
NAMD 3.0 alpha 6 and later add support for GPU-accelerated FEP/TI simulations

- **NAMD_3.0alpha8_Linux-x86_64-multicore-CUDA-SingleNode.tar.gz** (Standard simulation.) **(December 21, 2020)**
- **NAMD_3.0alpha8_Linux-x86_64-netlrts-smp-CUDA-SingleNode.tar.gz** (Multi-copy GPU simulations. For multi-GPU machines use: "+devicesperreplica 1") **(December 21, 2020)**
- **NAMD_3.0alpha7_Linux-x86_64-multicore-CUDA-SingleNode.tar.gz** (Standard simulation.) **(October 16, 2020)**
- **NAMD_3.0alpha7_Linux-x86_64-netlrts-smp-CUDA-SingleNode.tar.gz** (Multi-copy GPU simulations. For multi-GPU machines use: "+devicesperreplica 1") **(October 16, 2020)**
- **NAMD_3.0alpha6_Linux-x86_64-multicore-CUDA-SingleNode.tar.gz** (Standard simulation.) **(August 12, 2020)**
- **NAMD_3.0alpha6_Linux-x86_64-netlrts-smp-CUDA-SingleNode.tar.gz** (Multi-copy GPU simulations. For multi-GPU machines use: "+devicesperreplica 1") **(August 12, 2020)**

NAMD 3.0 alpha 4 and later automatically manage stepsPerCycle and pairlistsPerCycle

- **NAMD_3.0alpha5_Linux-x86_64-multicore-CUDA-SingleNode.tar.gz** (Standard simulation.) **(July 22, 2020)**
- **NAMD_3.0alpha5_Linux-x86_64-netlrts-smp-CUDA-SingleNode.tar.gz** (Multi-copy GPU simulations. For multi-GPU machines use: "+devicesperreplica 1") **(July 22, 2020)**
- **NAMD_3.0alpha4_Linux-x86_64-multicore-CUDA-SingleNode.tar.gz** (Standard simulation.) **(July 18, 2020)**
- **NAMD_3.0alpha4_Linux-x86_64-netlrts-smp-CUDA-SingleNode.tar.gz** (Multi-copy GPU simulations. For multi-GPU machines use: "+devicesperreplica 1") **(July 18, 2020)**

NAMD 3.0 alpha 3 and earlier need stepsPerCycle and pairlistsPerCycle tuned, e.g., to 400, and 40 respectively, for best performance



```
[5]  charmm-gui-6838781093/toppar/par_all3
     charmm-gui-6838781093/toppar/toppar_a
     charmm-gui-6838781093/toppar/toppar_a
```

```
[8]  %cd /content/charmm-gui-68387810
     /content/charmm-gui-6838781093/namd
```

Modify namd2 path in README, su

/content/NAMD_Git-**_Linux-x86_

Start a normal simulation (skip thi

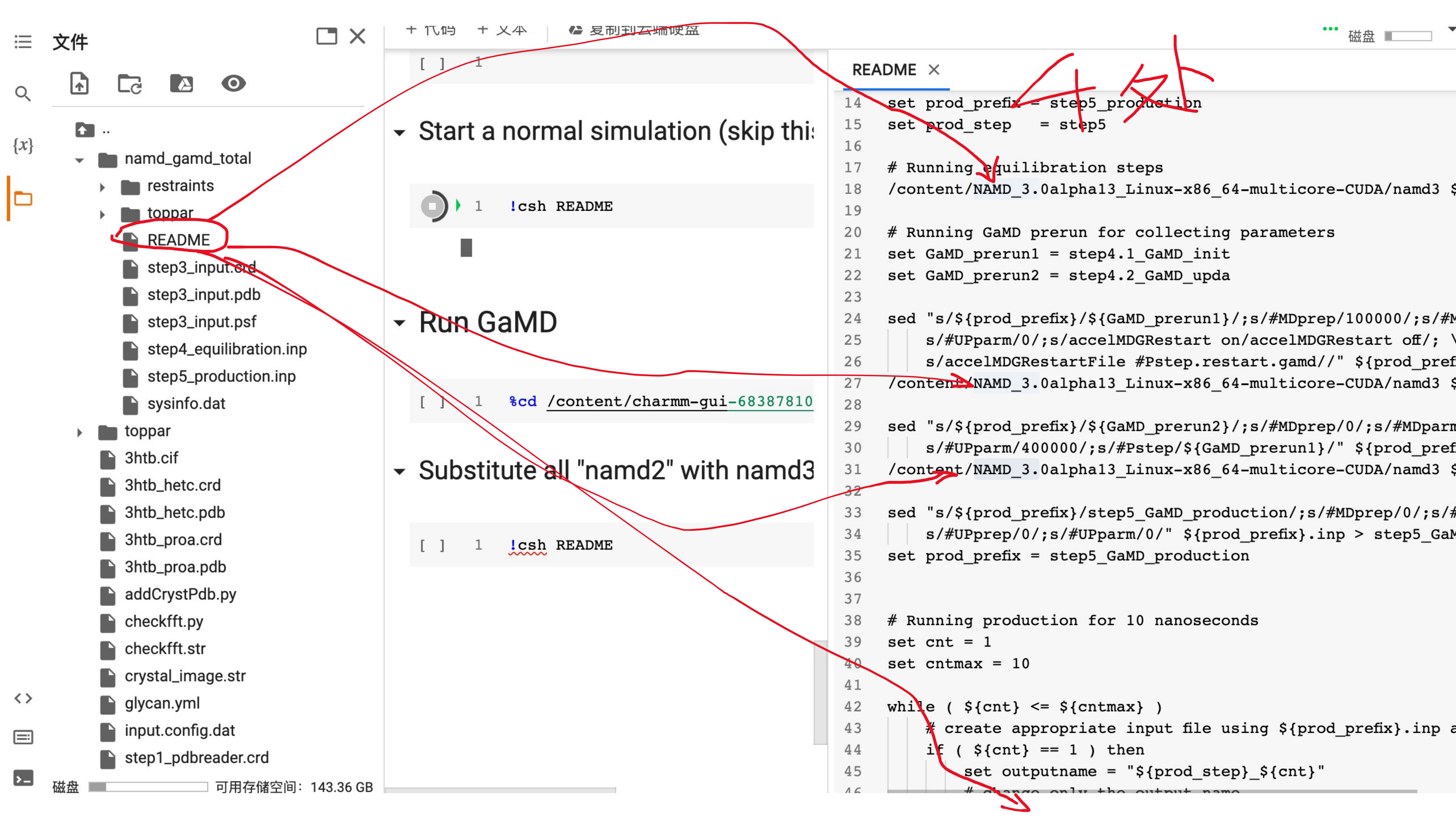
```
[ ]  !csh README
```

Run GaMD

```
[ ]  %cd /content/charmm-gui-68387810
```

Substitute all "namd2" with namd3

```
set equi_prefix = step4_equilibration
set prod_prefix = step5_production
set prod_step   = step5

# Running equilibration step
/content/NAMD_3.0alpha13_Linux-x86_64-multicore-CUDA/namd3

# Running production for 10 nanoseconds
set cnt    = 1
set cntmax = 10

while ( ${cnt} <= ${cntmax} )
    # create appropriate input file using ${prod_prefix}.inp
    if ( ${cnt} == 1 ) then
        set outputname = "${prod_step}_${cnt}"
        # change only the output name
        sed "s/${prod_prefix}/${outputname}/" ${prod_prefix}
    else
        @ cntprev = ${cnt} - 1
        set inputname  = "${prod_step}_${cntprev}"
        set outputname = "${prod_step}_${cnt}"
        # change input and output names from template file
        sed "s/${equi_prefix}/${inputname}/" ${prod_prefix}.
            sed "s/${prod_prefix}/${outputname}/" > ${prod_
    endif

    # run the simulation for 1 nanosecond
    /content/NAMD_3.0alpha13_Linux-x86_64-multicore-CUDA/n

    @ cnt += 1
end
```



```
..
namd_gamd_total
  restraints
  toppar
  README
  step3_input.crd
  step3_input.pdb
  step3_input.psf
  step4_equilibration.inp
  step5_production.inp
  sysinfo.dat
toppar
3htb.cif
3htb_hetc.crd
3htb_hetc.pdb
3htb_proa.crd
3htb_proa.pdb
addCrystPdb.py
checkfft.py
checkfft.str
crystal_image.str
glycan.yml
input.config.dat
step1_pdbreader.crd
```

磁盘 可用存储空间：143.36 GB

Start a normal simulation (skip thi

```
!csh README
```

Run GaMD

```
%cd /content/charmm-gui-68387810
```

Substitute all "namd2" with namd3

```
!csh README
```

README

```
set prod_prefix = step5_production
set prod_step   = step5

# Running equilibration steps
/content/NAMD_3.0alpha13_Linux-x86_64-multicore-CUDA/namd3 $

# Running GaMD prerun for collecting parameters
set GaMD_prerun1 = step4.1_GaMD_init
set GaMD_prerun2 = step4.2_GaMD_upda

sed "s/${prod_prefix}/${GaMD_prerun1}/;s/#MDprep/100000/;s/#M
    s/#UPparm/0/;s/accelMDGRestart on/accelMDGRestart off/; \
    s/accelMDGRestartFile #Pstep.restart.gamd//" ${prod_pref
/content/NAMD_3.0alpha13_Linux-x86_64-multicore-CUDA/namd3 $

sed "s/${prod_prefix}/${GaMD_prerun2}/;s/#MDprep/0/;s/#MDparm
    s/#UPparm/400000/;s/#Pstep/${GaMD_prerun1}/" ${prod_pref
/content/NAMD_3.0alpha13_Linux-x86_64-multicore-CUDA/namd3 $

sed "s/${prod_prefix}/step5_GaMD_production/;s/#MDprep/0/;s/#
    s/#UPprep/0/;s/#UPparm/0/" ${prod_prefix}.inp > step5_GaM
set prod_prefix = step5_GaMD_production


# Running production for 10 nanoseconds
set cnt = 1
set cntmax = 10

while ( ${cnt} <= ${cntmax} )
    # create appropriate input file using ${prod_prefix}.inp a
    if ( ${cnt} == 1 ) then
        set outputname = "${prod_step}_${cnt}"
        # change only the output name
```