Molecular Dynamics Simulations

Oleg N. Starovoytov

HPC User Services

LSU HPC / LONI

sys-help@loni.org

Louisiana State University

Baton Rouge

November 13, 2024

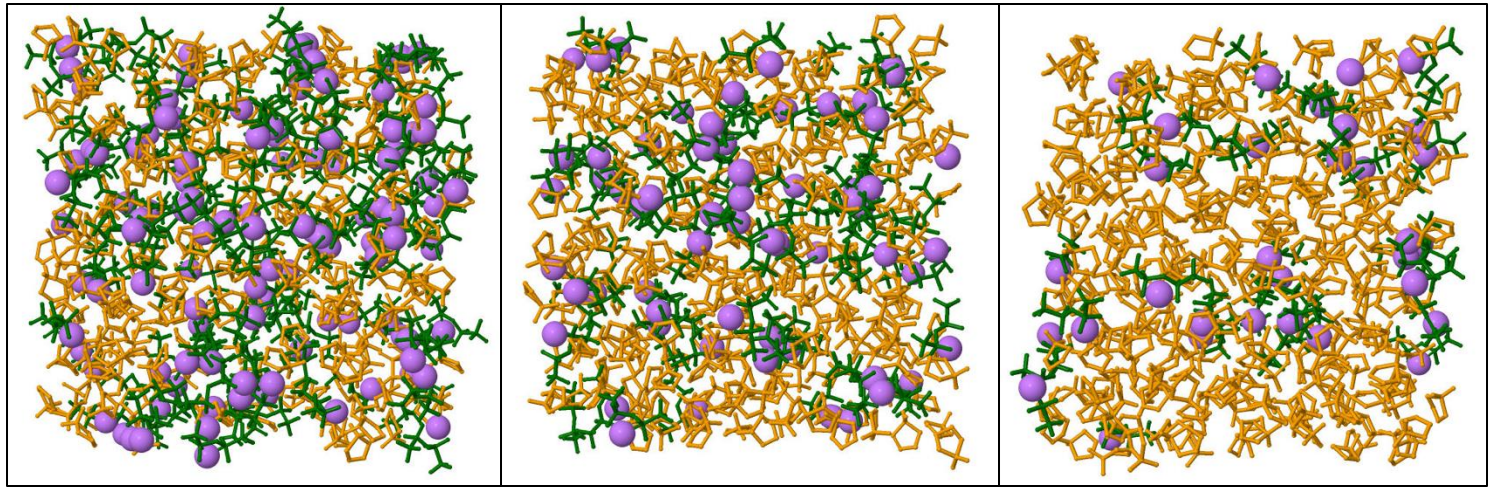
Outline

- **Part 1**

  1. Introduction to Molecular Dynamics Simulations
  2. Molecular Dynamics Simulation packages
  3. HPC LSU and LONI Software environment

- **Part 2**

  1. Running MD simulations using available packages on an HPC System



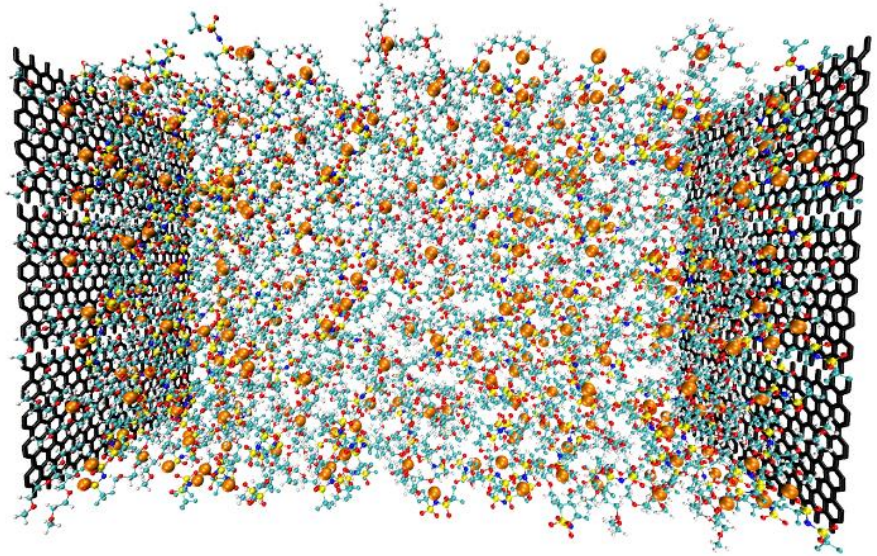
LiTFSI: 2.17 mol/dm^3 LiTFSI: 1.77 mol/dm^3 LiTFSI: 1.03 mol/dm^3

Introduction to Molecular Dynamics Simulations

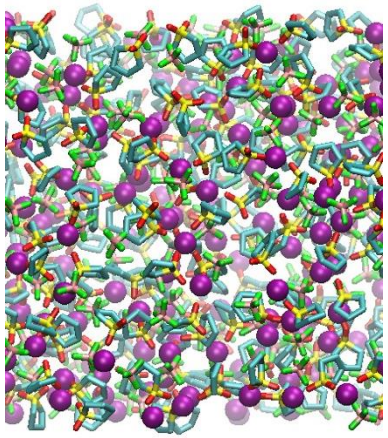
Introduction to Molecular Dynamics Simulations

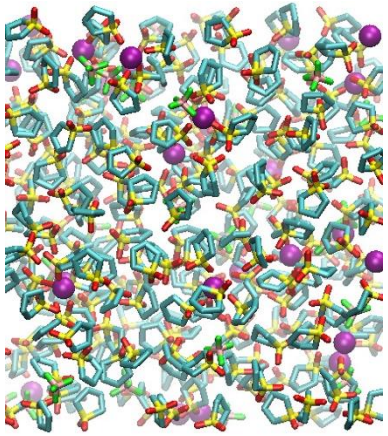
Molecular dynamics (MD) is a versatile tool for calculating structural and dynamics properties of molecular systems at equilibrium as a function of time. The molecular systems should obey the laws of classical physics.

Molecular dynamics simulations are widely used in various fields like chemistry, physics, biology, and material science. The most popular research studies include the dynamics of proteins, DNA and RNA structures, ionic liquids (ILs), lipid bilayers and membranes, battery electrolytes, and …



Ion dynamics in battery electrolytes





Introduction to Molecular Dynamics Simulations

Molecular dynamics simulation models

1. Ensemble of atoms, each has a point mass m.
2. Group of atoms (OPLS-UA)
3. Coarse-grained models (MARTINI model)
4. Machine learning models

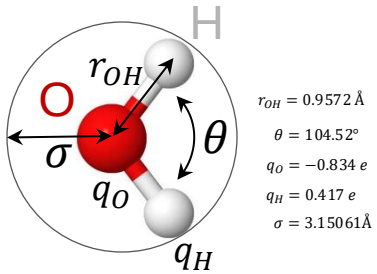
Democritus ~400 B.C.

The Greek word *ατομοσ* means invisible.

Force fields

1. Pair-wise classical force fields (AMBER, CHARMM, OPLS, GROMOS)
2. Many-body force fields include (EAM, Tersoff, REBO)
3. Reactive force fields (ReaxFF)
4. Machine learning force fields

TIP3P water model



Molecular dynamics simulations

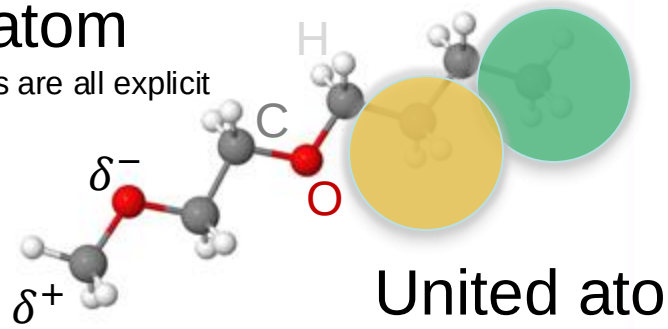
1. Integrate Newton's equation of motion
2. F = ma
3. Set 3N ODEs to propagate over time (Velocity Verlet Algorithm)

Thermodynamic properties

1. Calculate structural and dynamic properties as a time average of an ensemble of atoms.

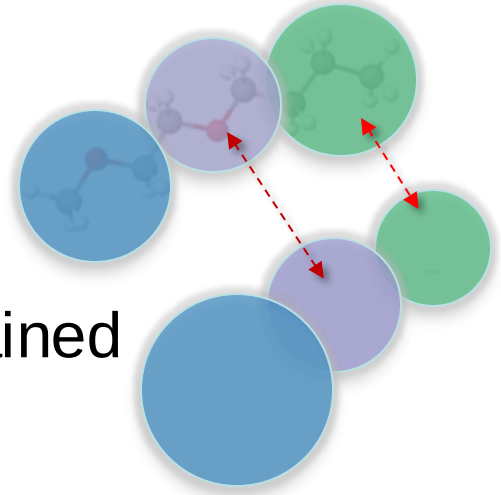
All atom

H atoms are all explicit



United atom

Beads include H atoms in CH2 CH3

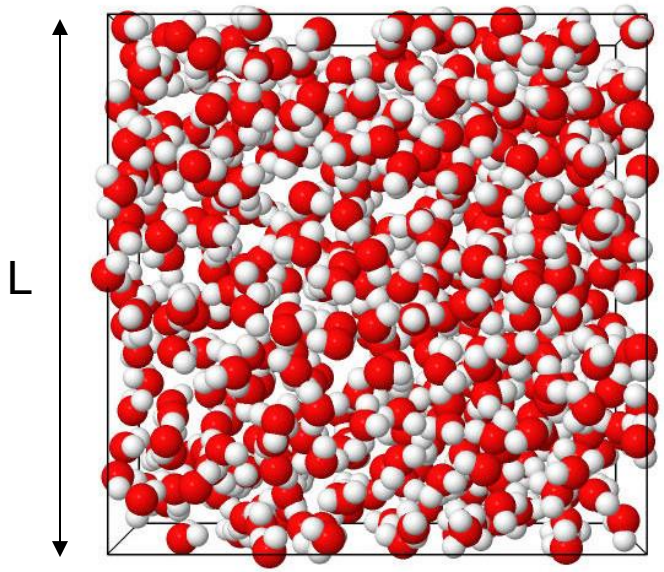


Coarse-Grained models

Beads include entire functional groups

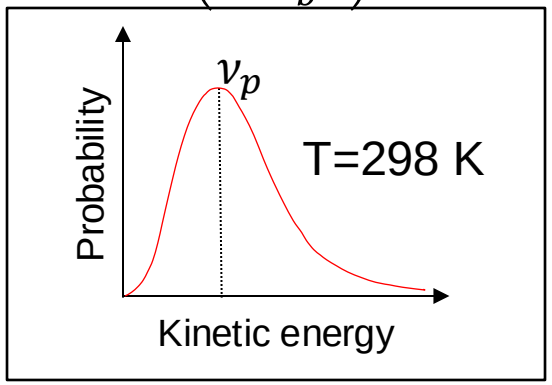
Introduction to Molecular Dynamics Simulations

Simulation box, L



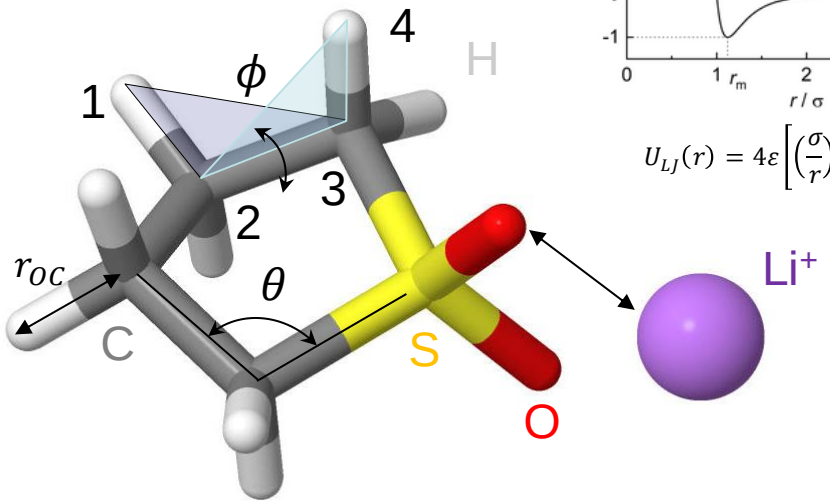
$N = 1536 \quad r_i = x, y, z$

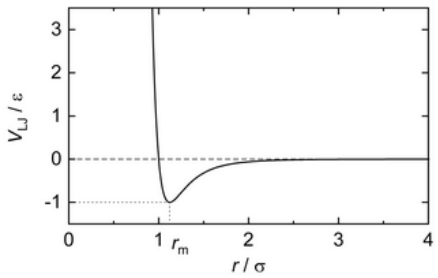
$$P(v_{i,r}) = \left(\frac{m}{2\pi k_b T}\right)^{\frac{1}{2}} e^{-\frac{m v_{i,r}^2}{2\pi k_b T}}$$



1. Set up a system of N atoms.
2. Assign x, y, and z coordinates to each atom
3. Assign velocities using Maxwell-Boltzmann Distribution
4. Choose the right force field (Potential function)
5. Propagate atomic positions using integration algorithms (Velocity Verlet, Leap Frog, and …)

Force Field





$$U_{LJ}(r) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

$$U_{total} = U_{intra} + U_{inter}$$

$$U_{intra} = U_{bond} + U_{bend} + U_{torsion} + U_{out\ of\ plane}$$

$$U_{inter} = U_{Coulomb} + U_{van\ der\ Waals}$$

Introduction to Molecular Dynamics Simulations

Time step: Δt

Intra-molecular interactions	Neighbor list	Inter-molecular Interactions: van der Waals (LJ), Electrostatic (Ewald)	Calculate forces, update positions, adjust temperature and pressure

$$r(t+\Delta t) = r(t) + \Delta t v(t) + \frac{\Delta t^2 a(t)}{2}$$

$$a(t+\Delta t) = \frac{f(t+\Delta t)}{m}$$

$$v(t+\Delta t) = v(t) + \frac{1}{2}\Delta t[a(t) + a(t+\Delta t)]$$

Introduction to Molecular Dynamics Simulations

1. Ewald summation (by Peter P. Ewald in 1921)
2. PPME (Particle-Particle mesh Ewald, Hockney 1981)
3. PME (Particle mesh Ewald, Darden 1993)

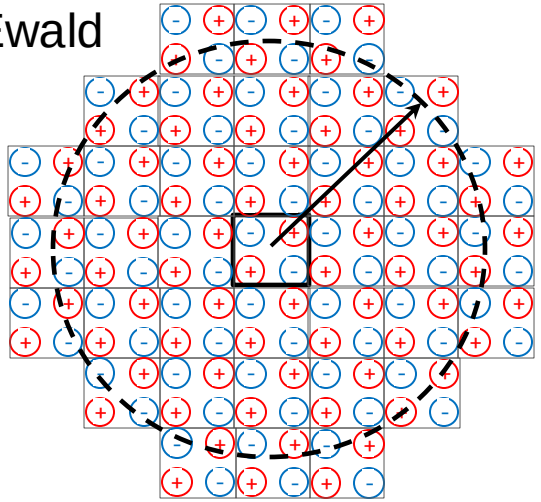
$$U^{Ewald} = U^{real} + U^{reciprocal} + U^{self}$$

$\mathcal{O}(N^{3/2})$ → $\mathcal{O}(N \cdot log(N))$

A three-dimensional grid is introduced to optimize the computation of long-range interactions and calculate the reciprocal space contribution.

A discrete set of points is introduced where charge densities and potentials are calculated significantly reducing the number of calculations needed for reciprocal space.

Ewald



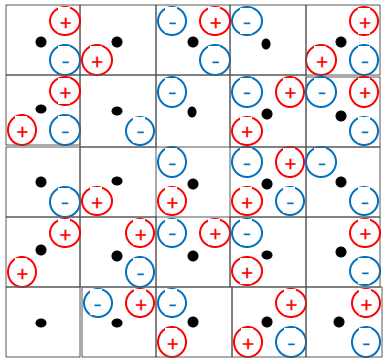
$$\sum_{i=1}^{N} q_i = 0$$

$$\mathcal{O}(N)$$

$$U^{real} = \frac{1}{2}\sum_{i,j}^{N}\sum_{n} q_i q_j \frac{\mathrm{erfc}(\alpha r_{ij,n})}{r_{ij}}$$

$$U^{self} = -\frac{\alpha}{\sqrt{\pi}}\sum_{i=1}^{N} q_i^2$$

$$U^{reciprocal} = \frac{1}{2\pi V}\sum_{i,j}^{N} q_i q_j \sum_{k\neq 0} \frac{exp\left(-\frac{\pi k^2}{\alpha}\right) + 2\pi i k \cdot (r_i - r_j)}{k^2}$$



PME

$$\mathcal{O}(N \cdot log(N))$$

Molecular Dynamics Simulation Packages

MD Simulation Packages

- LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator, Sandia National Lab, 1995) is a molecular dynamics simulation package, https://www.lammps.org

- GROMACS (GROningen Machine for Chemical Simulations, University of Groningen, 1991) is a molecular dynamics simulation package, https://www.gromacs.org

- NAMD (Not Another Molecular Dynamics Program, University of Illinois Urbana-Champaign, 1995) is a molecular dynamics simulation package (CHARMM force field), https://www.ks.uiuc.edu/Research/namd

- AMBER (Assisted Model Building with Energy Refinement, University of California, 2002) is a molecular dynamics simulation package (DNA force fields), https://ambermd.org

MD Simulation Packages

Name	Model builder	Min	MD	MC	GPU	License
LAMMPS	Yes	Yes	Yes	Yes	Yes	Free
GROMACS	No	Yes	Yes	No	Yes	Free
NAMD	Yes	Yes	Yes	No	Yes	Free
AMBER	Yes	Yes	Yes	Yes	Yes	Proprietary

MD Simulation Packages

	LAMMPS	GROMACS	NAMD	AMBER	VMD
QBC	2020/03/03	2020.2	2.14	18/22	1.9.3
QBD	2023/08/02	2021.7	2.14	22	1.9.3
SMIC	2022/12/22	2022.0	2.14	18	1.9.3
MIKE	2022/06/23	2021.3	2.14	22	1.9.3

HPC LSU and LONI Software Environment

Software environment

```
[user@mike2 ~]$ module av

amber/18/intel-2021.5.0-intel-mpi-2021.5.1
amber/22/intel-2021.5.0-cuda-11.5.0-intel-mpi-2021.5.1
amber/22/intel-2021.5.0-intel-mpi-2021.5.1
.
gromacs/2021.3/intel-2021.5.0-intel-mpi-2021.5.1
.
lammps/02Aug2023/intel-2021.5.0-cuda-11.6.0-intel-mpi-2021.5.1
lammps/02Aug2023/intel-2021.5.0-intel-mpi-2021.5.1
.
namd/2.14/intel-2021.5.0
namd/2.14/intel-2021.5.0-cuda
```

Software environment

No GPU

lammps/02Aug2023/intel-2021.5.0-intel-mpi-2021.5.1

name | package version | compiler | MPI library

lammps/02Aug2023/intel-2021.5.0-cuda-11.6.0-intel-mpi-2021.5.1

name | package version | compiler | cuda version | MPI library

GPU

Software environment

[user@mike2 ~]$ **module purge**

[user@mike2 ~]$ **module load** lammps/23Jun2022/intel-2021.5.0-intel-mpi-2021.5.1

[user@mike2 ~]$ **module list**

Currently Loaded Module files:

1) intel/2021.5.0
2) intel-mpi/2021.5.1
3) lammps/23Jun2022/intel-2021.5.0-intel-mpi-2021.5.1

Software environment

```
[user@mike2 ~]$ module display lammps/02Aug2023/intel-2021.5.0-intel-mpi-2021.5.1

module-whatis   {LAMMPS stands for Large-scale Atomic/Molecular Massively Parallel Simulator. This package uses patch releases, not stable release. See https://github.com/spack/spack/pull/5342 for a detailed discussion. }

conflict        lammps

prepend-path    PATH /usr/local/packages/lammps/02Aug2023/intel-2021.5.0-intel-mpi-2021.5.1/bin

prepend-path    MANPATH /usr/local/packages/lammps/02Aug2023/intel-2021.5.0-intel-mpi-2021.5.1/share/man
```

Software environment

```
[user@mike2 ~]$ ls /usr/local/packages/lammps/02Aug2023/intel-2021.5.0-intel-mpi-2021.5.1/bin

binary2txt  chain.x  lmp_mpi  micelle2d.x  msi2lmp  phana  stl_bin2txt
```

Running jobs on HPC

In **High-Performance Computing (HPC)** environments, there are two types of jobs.

1. Interactive Jobs

Interactive jobs allow the user to run tasks interactively, usually with a direct connection to the system through a terminal or GUI. These jobs are intended for tasks that require interaction from the user.

salloc and/or srun commands

2. Batch Jobs

Batch jobs are computational tasks that are submitted to an HPC cluster without the need for interaction from the user. These jobs are queued and executed by the **slurm** manager when resources are available.

sbatch <script.sh>

Running Molecular Dynamics Simulations Using Available Packages

Running LAMMPS jobs on HPC

Every LAMMPS simulation needs two essential files:

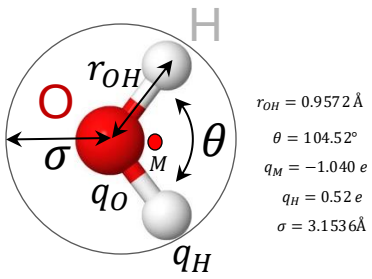
1. structure / topology (.lammps),
2. parameters (.in).

Structure/topology

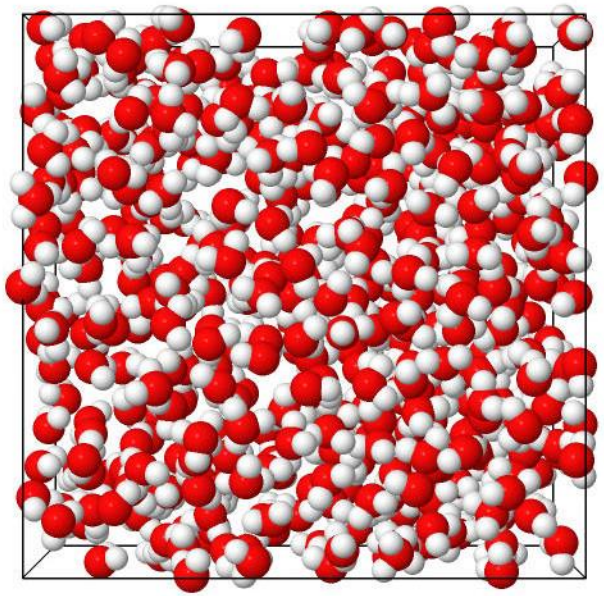
```
1536 atoms
Atoms
  1  1  1 -1.04840  23.067397  25.992172  12.516813
  2  1  2  0.52420  23.651513  25.756170  13.277936
  3  1  2  0.52420  23.106625  25.196754  11.981115
  ...
```

Parameters

```
units         real
atom_style    full
boundary      p p p

# Force Field
pair_style    lj/cut/tip4p/cut 1 2 1 1 0.125 8.0
bond_style    harmonic
angle_style   harmonic
kspace_style  none

#Read data
read_data     tip4p_512.lammps
...
```



TIP4P water model



Running LAMMPS jobs on HPC

Running LAMMPS interactively

```
[user@mike2 LAMMPS]$ srun   -N1 -n64 -p workq --time=05:00:00 -A  hpc_allocation --pty bash
[user@mike2 LAMMPS]$ salloc -N1 -n64 -p workq --time=05:00:00 -A  hpc_allocation
[user@mike2 LAMMPS]$ module purge
[user@mike2 LAMMPS]$ module load lammps/02Aug2023/intel-2021.5.0-intel-mpi-2021.5.1
[user@mike2 LAMMPS]$ module list

Currently Loaded Modulefiles:
 1) intel/2021.5.0   2) intel-mpi/2021.5.1   3) lammps/02Aug2023/intel-2021.5.0-intel-mpi-2021.5.1

[user@mike171 LAMMPS]$ srun --overlap –n 1  lmp_mpi -in tip4p_512.in > tip4p_512.out & (for srun)
[user@mike171 LAMMPS]$ srun         –n 1  lmp_mpi -in tip4p_512.in > tip4p_512.out & (for salloc)

[user@mike171 LAMMPS]$ srun --overlap –n 64 lmp_mpi -in tip4p_512.in > tip4p_512.out & (for srun)
[user@mike171 LAMMPS]$ srun         –n 64 lmp_mpi -in tip4p_512.in > tip4p_512.out & (for salloc)

log.lammps  tip4p_512.in  tip4p_512.lammps  tip4p_512.out  tip4p_512.traj
```

To check the processes running for a particular user.

username@mike0XX $ **ps -u username –f**

To kill the processes running for a particular user.

username@mike0XX $ **pkill -u username**

If you reserve an interactive node using **salloc** command, you do not need to include the **- - overalp** option while running the job.

Running LAMMPS jobs on HPC

Running LAMMPS jobs using PBS system

```
#!/bin/bash
#PBS -q workq
#PBS -N test
#PBS -l nodes=1:ppn=20
#PBS -l walltime=HH:MM:SS
#PBS -A loni_allocation
#PBS -o lammps.${PBS_JOBID}.out
#PBS -e lammps.${PBS_JOBID}.err
#PBS -m bea
#PBS -M your@email.address

module purge
module load lammps/20201029/intel-19.0.5-cuda-mvapich-2.3.3

echo "Date            = $(date)"
echo "Hostname        = $(hostname -s)"
echo "Working directory  = $(pwd)"

echo $PBS_O_WORKDIR
cd   $PBS_O_WORKDIR

time mpirun -np 20 lmp_mpi -in tip4p_512.in \
> tip4p_512.out
```

Running LAMMPS jobs using SLURM system

```
#!/bin/bash
#SBATCH -p workq
#SBATCH -N 1
#SBATCH -n 64
#SBATCH -c 1
#SBATCH -t HH:MM:SS
#SBATCH -A hpc_allocation
#SBATCH –J job_name
#SBATCH -o lammps.%j.out
#SBATCH -e lammps.%j.err
#SBATCH --mail-user=your@email.address
#SBATCH --mail-type=END,FAIL
module purge
module load module load lammps/23Jun2022/intel-2021.5.0-intel-mpi-2021.5.1

echo “JOBID NUMBER: “      $SLURM_JOBID
echo “NUMBER OF NODES: “   $SLURM_NNODES
echo “WORKING DIRECTORY: “ $SLURM_SUBMIT_DIR
echo “NODE LIST: “         $SLURM_JOB_NODELIST

cd   $SLURM_SUBMIT_DIR

time srun -N1 -n64 lmp -in water_tip4p.in > water_tip4p.out
```

Running GROMACS jobs on HPC

Every GROMACS simulation needs three essential files:

1. structure (.gro/.pdb),
2. topology (.top), and
3. parameters (.mdp).

Structure

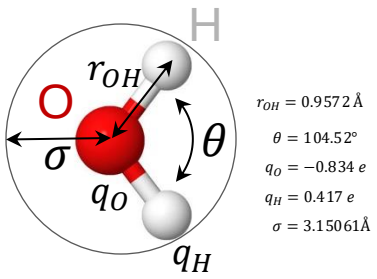
```
TIP3P water
 1536
   1SOL    OW   1  2.308  1.150  1.290  0.0374 -0.1946  0.1896
   1SOL   HW1   2  2.242  1.208  1.328 -0.7293 -1.1860  0.4498
   1SOL   HW2   3  2.376  1.143  1.357  1.6678  2.7051 -0.9958
   ...
```

Topology

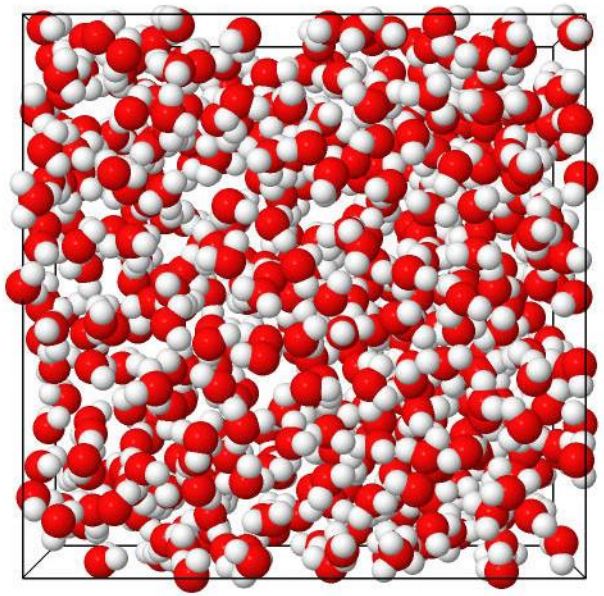
```
; Include forcefield parameters
#include "charmm27.ff/forcefield.itp"

; Include water topology
#include "charmm27.ff/tip3p.itp”
...
```

Parameters

```
; Run parameters
integrator          = md    ;leap-frog integrator
nsteps              = 5000 ;
dt                  = 0.002 ;
...
```



TIP3P water model



Running GROMACS jobs on HPC

Running GROMACS interactively

```
[user@mike2 GROMACS]$ srun -N1 -n64 -p workq --time=05:00:00 -A  hpc_allocation0 --pty bash
[user@mike2 GROMACS]$ module purge
[user@mike2 GROMACS]$ module load gromacs/2021.3/intel-2021.5.0-intel-mpi-2021.5.1
[user@mike2 GROMACS]$ module list

Currently Loaded Modulefiles:
 1) intel/2021.5.0   2) intel-mpi/2021.5.1   3) gromacs/2021.3/intel-2021.5.0-intel-mpi-2021.5.1

[user@mike2 GROMACS]$ srun --overlap gmx_mpi grompp  -f  min.mdp -c npt.gro -p topol.top -o min.tpr
[user@mike2 GROMACS]$ srun --overlap gmx_mpi  mdrun --deffnm min
[user@mike2 GROMACS]$ ls

min.edr  min.gro  min.log  min.mdp  min.tpr  min.trr

[user@mike2 GROMACS]$ srun --overlap gmx_mpi grompp  -f  eql.mdp -c min.gro -p topol.top -o eql.tpr
[user@mike2 GROMACS]$ srun --overlap gmx_mpi  mdrun --deffnm eql
[user@mike2 GROMACS]$ ls

eql.cpt  eql.edr  eql.gro  eql.log  eql.mdp  eql.tpr  eql.trr  eql.xtc
```

Running GROMACS jobs on HPC

Running GROMACS jobs using PBS system

```
#!/bin/bash
#PBS -q workq
#PBS –N test
#PBS -l nodes=1:ppn=20
#PBS -l walltime=HH:MM:SS
#PBS -A loni_allocation
#PBS -o gromacs.${PBS_JOBID}.out
#PBS -e gromacs.${PBS_JOBID}.err
#PBS -m bea
#PBS -M your@email.address

module purge
module load gromacs/2020.6/intel-19.0.5-mvapich-2.3.3


echo "Date            = $(date)"
echo "Hostname        = $(hostname -s)"
echo "Working Directory = $(pwd)"

echo $PBS_O_WORKDIR
cd   $PBS_O_WORKDIR

time mpirun -np 20 gmx_mpi mdrun –deffnm npt -v
```

Running GROMACS jobs using SLURM system

```
#!/bin/bash
#SBATCH -p workq
#SBATCH -N 1
#SBATCH -n 64
#SBATCH -c 1
#SBATCH -t HH:MM:SS
#SBATCH -A hpc_allocation
#SBATCH –J test
#SBATCH -o gromacs.%j.out
#SBATCH -e gromacs.%j.err
#SBATCH --mail-user=your@email.address

module purge
module load gromacs/2021.3/intel-2021.5.0-intel-mpi-2021.5.1

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

echo $SLURM_JOBID
echo $SLURM_NNODES
echo $SLURM_NTASKS
echo $SLURM_SUBMIT_DIR

cd   $SLURM_SUBMIT_DIR

time srun -N1 -n64 gmx_mpi mdrun –deffnm eql -v
```

Running NAMD jobs on HPC

Every NAMD simulation needs three essential files:
Input files are identical to the input files used by **X-PLOR** and **CHARMM**.

1. coordinates (.pdb),
2. structure (.psf),
3. topology (.xplor),
4. parameters (.namd).

Coordinates (.pdb)

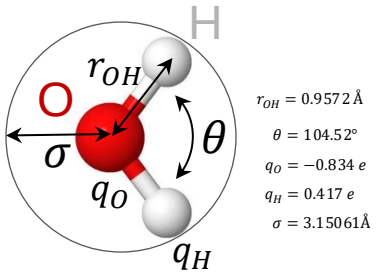
```
REMARK original generated coordinate pdb file
ATOM    1  OH2 TIP3W  5     3.668  10.082  15.904  1.00  0.00     WW1  O
ATOM    2  H1  TIP3W  5     3.224  10.451  15.101  1.00  0.00     WW1  H
ATOM    3  H2  TIP3W  5     3.092  10.379  16.627  1.00  0.00     WW1  H
...
```

Structure (.psf)

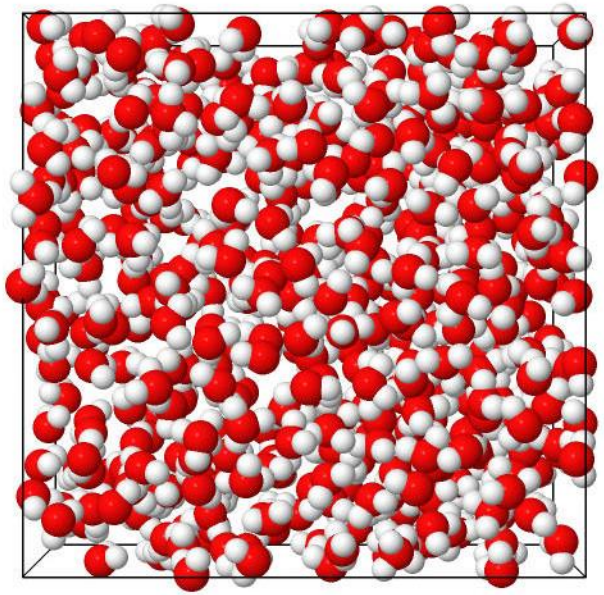
```
  1536 !NATOM
    1 WW1  5   TIP3 OH2  OT   -0.834000     15.9994        0
    2 WW1  5   TIP3 H1   HT    0.417000     1.0080         0
    3 WW1  5   TIP3 H2   HT    0.417000     1.0080         0
  ...
```

Topology (.xplor)

```
 BOND  OT   HT   450.000    0.9572 ! ALLOW   WAT
 BOND  HT   HT     0.000    1.5139 ! ALLOW WAT
ANGLE  HT   OT   HT     55.000 104.5200  ! ALLOW WAT
...
```



TIP3P water model



Parameters (.input)

```
timestep            1.0
fullElectFrequency   4
numsteps            50000
outputtiming         20
...
```

Running NAMD jobs on HPC

Running NAMD interactively

```
[user@mike2 NAMD]$ srun -N1 -n64 -p workq --time=05:00:00 -A  hpc_allocation0 --pty bash
[user@mike2 NAMD]$ module purge
[user@mike2 NAMD]$ module load namd/2.14/intel-2021.5.0
[user@mike2 NAMD]$ module list

Currently Loaded Modulefiles:
 1) intel/2021.5.0   2) namd/2.14/intel-2021.5.0

[user@mike2 NAMD]$ srun --overlap namd2 tip3p_512.namd > tip3p_512.out &
[user@mike2 NAMD]$ ls

par_all22_prot_lipid.xplor  tip3p_512.out  tip3p_512.out.coor.BAK  tip3p_512.out.vel.BAK
tip3p_512.out.xsc.BAK  tip3p_512.psf   tip3p_512.nam p3p_512.out.coor  tip3p_512.out.vel tip3p_512.out.xsc      tip3p_512.pdb
```

Running NAMD jobs on HPC

Running NAMD jobs using PBS system

```
#!/bin/bash
#PBS -q workq
#PBS -N test
#PBS -l nodes=1:ppn=20
#PBS -l walltime=HH:MM:SS
#PBS -A loni_allocation
#PBS -e NAMD.${PBS_JOBID}.err
#PBS -o NAMD.${PBS_JOBID}.out
#PBS -m bea
#PBS -M your@email.address

module purge
module load namd/2.14/intel-19.0.5

echo "Date            = $(date)"
echo "Hostname        = $(hostname -s)"
echo "Working Directory = $(pwd)"

echo $PBS_O_WORKDIR
cd   $PBS_O_WORKDIR

time mpirun -np 20 namd2 tip3p_512.namd > tip3p_512.out
```

Running NAMD jobs using SLURM system

```
#!/bin/bash
#SBATCH -p workq
#SBATCH -N 1
#SBATCH -n 64
#SBATCH -c 1
#SBATCH -t HH:MM:SS
#SBATCH -A hpc_allocation
#SBATCH –J test
#SBATCH -o NAMD.%j.out
#SBATCH -e NAMD.%j.err
#SBATCH --mail-user=your@email.address

module purge
module load namd/2.14/intel-2021.5.0

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

echo $SLURM_SUBMIT_DIR
cd   $SLURM_SUBMIT_DIR


time srun -N1 -n64 namd2 tip3p_512.namd > tip3p_512.out
```

Running AMBER jobs on HPC

Every AMBER simulation needs three essential files: Initial coordinate, topology, and parameter files.

1. coordinates (.inpcrd),
2. topology (.prmtop),
3. parameter (.inp).

Structure (.inpcrd)

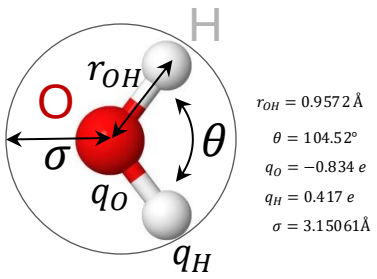
```
default_name
 1536
  16.5307255  19.4975686  18.1539268  13.3987255  16.6285686  12.9069268
  15.2747255  15.8905686  11.7989268  17.0747255  16.7645686  10.4629268
...
```

Topology (.prmtop)

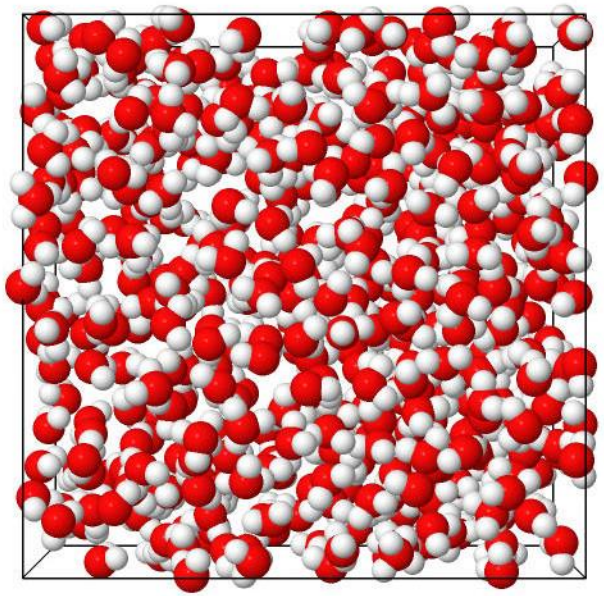
```
 %FORMAT(10I8)
   1536      2   1024      0    512      0  ...
   2048    512      0      0      0      2  ...
      0      0      0      0      0      0  ...
 ...
```

Parameters (.inp)

```
Production
 &cntrl
  imin=0,
  ntx=1,
  ntwv = 1
...
```



TIP3P water model



Running AMBER jobs on HPC

Running AMBER interactively

```
[user@mike2 AMBER]$ srun -N1 -n64 -p workq --time=05:00:00 -A  hpc_allocation0 --pty bash
[user@mike2 AMBER]$ module purge
[user@mike2 AMBER]$ module load amber/22/intel-2021.5.0-intel-mpi-2021.5.1
[user@mike2 AMBER]$ module list

Currently Loaded Modulefiles:
  1) intel/2021.5.0   2) intel-mpi/2021.5.1   3) amber/22/intel-2021.5.0-intel-mpi-2021.5.1

[user@mike2 AMBER]$ srun --overlap sander -O -i eql.inp -o eql.out -p tip3p_512.prmtop -c tip3p_512.inpcrd &
[user@mike2 AMBER]$ ls

[user@mike2 AMBER]$ srun --overlap –n1 –N64 sander.MPI -O -i eql.inp -o eql.out -p tip3p_512.prmtop -c tip3p_512.inpcrd &
[user@mike2 AMBER]$ ls


[user@mike2 AMBER]$ srun --overlap –n1 –N64  pmemd.MPI -O -i eql.inp -o eql.out -p tip3p_512.prmtop -c tip3p_512.inpcrd &
[user@mike2 AMBER]$ ls
```

Running AMBER jobs on HPC

Running AMBER jobs using PBS system

```
#!/bin/bash
#PBS -q workq
#PBS -N test
#PBS -l nodes=1:ppn=20
#PBS -l walltime= HH:MM:SS
#PBS -A loni_allocation
#PBS -o lammps.${PBS_JOBID}.out
#PBS -e lammps.${PBS_JOBID}.err
#PBS -m bea
#PBS -M your@email.address

module purge
module load amber/18/intel-19.0.5-mvapich-2.3.3

echo "Date            = $(date)"
echo "Hostname        = $(hostname -s)"
echo "Working directory  = $(pwd)"

echo $PBS_O_WORKDIR
cd   $PBS_O_WORKDIR

time mpirun -np 20 sander.MPI -O -i eql.inp \
 -o eql.out -p tip3p_512.prmtop \
 -c tip3p_512.inpcrd -r tip3p_512.rst
```

Running AMBER jobs using SLURM system

```
#!/bin/bash
#SBATCH -p workq
#SBATCH -N 1
#SBATCH -n 48
#SBATCH -c 1
#SBATCH -t HH:MM:SS
#SBATCH -A loni_allocation
#SBATCH -J test
#SBATCH -o amber.%j.out
#SBATCH -e amber.%j.err
#SBATCH --mail-user=your@email.address

module purge
module load amber/18/intel-19.0.5-mvapich-2.3.3

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

echo $SLURM_SUBMIT_DIR
cd   $SLURM_SUBMIT_DIR


time srun -N1 -n48 sander.MPI -O -i eql.inp -o eql.out -p tip3p_512.prmtop -c tip3p_512.inpcrd
-r tip3p_512.rst
```

- Marrink, S. J. *et al*, "The MARTINI Force Field: Coarse Grained Model for Biomolecular Simulations" J. Phys. Chem. B 2007, 111, 27, 7812-7824, https://doi.org/10.1021/jp071097f
- Van der Waals, Johannes Diderik (1837 - 1923). *Over de Continuiteit van den Gas- en Vloeistoftoestand.* Leiden, 1873, (Nobel Prize 1910, van der Waal's equation of state) http://rbx-exhibit2000.scs.illinois.edu//vanderwaals.htm
- Fabbrizzi, L. "Beyond the Molecule: Intermolecular Forces from Gas Liquefaction to X−H···π Hydrogen Bonds" ChemPlusChem, Volume 87, Issue 1, 2022, Pages 1-23, ISSN 2192-6506, https://doi.org/10.1002/cplu.202100243
- Darden, T. York, D. and Pederson, L. "Particle mesh Ewald: An $N \cdot \log(N)$ method for Ewald sums in large systems" J. Chem. Phys. 1993, 98, 10089-10092
- Plimpton, S. "Fast Parallel Algorithms for Short-Range Molecular Dynamics", Journal of Computational Physics, Volume 117, Issue 1, 1995, Pages 1-19, ISSN 0021-9991, https://doi.org/10.1006/jcph.1995.1039.
- https://stock.adobe.com/search?k=democritus&asset_id=620575197

Thank You

Summary of useful sinfo options

Usage: **sinfo** [OPTIONS]

Option	Description
-p <partition name>	Show nodes in specific partition
-t <state>	Filter nodes by state (e.g., idle, allocated, drain*)
-n <node name>	Show status for the specific node
-N	Display information about nodes

Examples

```
username@mike2 $ sinfo -p workq
username@mike2 $ sinfo -t idle
username@mike2 $ sinfo -n mike078
username@mike2 $ sinfo -N
```

Running jobs on HPC

```
username@mike2 $ sinfo

PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
single*     up 7-00:00:00      1 drain* mike077
checkpt     up 3-00:00:00     48  alloc mike[008-012,014]
checkpt     up 3-00:00:00    110   idle mike[001-007,114-123,128-159]
workq       up 3-00:00:00    110   idle mike[001-007]
bigmem      up 3-00:00:00      4   idle mike[172-175]
gpu2        up 3-00:00:00      1 drain* mike177
gpu4        up 3-00:00:00      1  alloc mike178

username@mike2 $ sinfo -p workq -t idle

PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
workq       up 3-00:00:00      3  drain mike[084,127,167]
workq       up 3-00:00:00    109   idle mike[001-007,019-028,039-059]
```

drain – the state that indicates maintenance of the node (not available to run jobs)
alloc – the state that indicates an allocated resource (not available to run jobs)
idle – the state that indicates an available resource (available to run jobs)

Running jobs on HPC

To run a job interactively on the allocated resources using **srun** in **Slurm**

Usage: **srun** [OPTIONS(0)...] [executable(0) [args(0)...]

Option	Description
-A <allocation name>	Charge job to specified allocation
-t <time>	Time limit: --time=DD-HH:MM:SS
-N <number of nodes>	Number of nodes: -NX
-n <number of cores>	Number of cores: -nXX
-p <partition>	Partition name: -p partition name

--gres=<resource> - specifies special resources like GPUs. (e.g. **--gres=gpu:2**)

Slurm will run a job on a compute node with one core for 12 hours on a single partition by default.

Examples

```
username@mike2 $ srun –A <Allocation> --pty bash
username@mike2 $ srun -p workq –A <Allocation> --pty bash
username@mike2 $ srun –N1 –n64 –p gpu2 -gres=gpu:2 –A <Allocation>
username@mike2 $ srun –N1 –n64 –p workq –nodelist=mikeXXX –A <Allocation>
```

You can use -w or --nodelist option when indicating a specific node.

Running jobs on HPC

To allocate resources on a **compute node** using **salloc** in **Slurm**

Usage: **salloc** [OPTIONS(0)...] [command(0) [args(0)...]

Option	Description
-A <allocation name>	Charge job to specified allocation
-t <time>	Time limit: --time=DD-HH:MM:SS
-N <number of nodes>	Number of nodes: -NX
-n <number of cores>	Number of cores: -nXX
-p <partition>	Partition name: -p partition name

--gres=<resource> - specifies special resources like GPUs. (e.g. **--gres=gpu:2**)

Slurm will run a job on a compute node with one core for 12 hours on a single partition by default.

Examples

```
username@mike2 $ salloc –A <Allocation>
username@mike2 $ salloc –N1 –n64 -p workq –A <Allocation>
username@mike2 $ salloc –w mike080 –p workq –A <Allocation>
username@mike2 $ salloc –p gpu2 –gres=gpu:2 –A <Allocation>
```

You can use -w or --nodelist option when indicating a specific node.

Running jobs on HPC

To check the job status and other related information, use the command

Usage: **scontrol show job [JOBID]**

Examples

```
username@mike2 $ scontrol show job 369822
JobId=369822 JobName=bash
   UserId=username(32428) GroupId=Admins(10000) MCS_label=N/A
   Priority=9352 Nice=0 Account=hpc_hpcadmin10 QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   RunTime=00:00:21 TimeLimit=12:00:00 TimeMin=N/A
   SubmitTime=2024-11-11T16:24:21 EligibleTime=2024-11-11T16:24:21
   StartTime=2024-11-11T16:24:22 EndTime=2024-11-12T04:24:22 Deadline=N/A
   Partition=workq AllocNode:Sid=mike2:293448
   NodeList=mike066
   NumNodes=1 NumCPUs=64 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   ReqTRES=cpu=1,mem=3920M,node=1,billing=1
   AllocTRES=cpu=64,mem=245G,node=1,billing=64
   MinCPUsNode=1 MinMemoryCPU=3920M MinTmpDiskNode=0
   Command=bash
   WorkDir=/ddnA/work/olegsupp/Training
   Comment=stdout=/ddnA/work/olegsupp/Training/slurm-369822.out
```

The <**scontrol show job**> command is valid for both types of jobs, **interactive** and **batch** jobs.

HPC Environment variables

Environment	PBS/Torque	Slurm
Job ID	$PBS_JOBID	$SLURM_JOBID
Submit directory	$PBS_O_WORKDIR	$SLURM_SUBMIT_DIR
Submit host	$PBS_O_HOST	$SLURM_SUBMIT_HOST
Node list	$PBS_NODEFILE	$SLURM_JOB_NODELIST
Total number of nodes	$PBS_NUM_NODES	$SLURM_NNODES
Total number of tasks	$PBS_NUM_PPN	$SLURM_NTASKS

PBS stands for Portable Batch System which provides a workload management system. It enables users to submit, monitor, and manage jobs on a supercomputer. It provides a framework for distributing computational tasks.

SLURM stands for Simple Linux Utility for Resource Management, which also provides a workload management system. It enables users to submit, monitor, and manage jobs on a supercomputer. It provides a framework for distributing computational tasks and a job scheduling system.

Running LAMMPS jobs on HPC with 1 GPU

Running LAMMPS interactively

```
[user@mike2 LAMMPS]$ srun -N1 -n16 -p gpu --gres=gpu:1 --time=12:00:00 -A hpc_hpcadmin8 --pty bash
[user@mike2 LAMMPS]$ module purge
[user@mike2 LAMMPS]$ module load lammps/02Aug2023/intel-2021.5.0-cuda-11.6.0-intel-mpi-2021.5.1
[user@mike2 LAMMPS]$ module list

Currently Loaded Modulefiles:
 1) intel/2021.5.0   2) intel-mpi/2021.5.1   3) lammps/02Aug2023/intel-2021.5.0-cuda-11.6.0-intel-mpi-2021.5.1


[user@mike179 LAMMPS]$ time srun --overlap -N1 –n16 lmp_mpi -sf gpu -pk gpu 1 neigh yes newton off -in lj.in > lj.out &

[user@mike179 LAMMPS]$ nvidia-smi –l
```

Running LAMMPS jobs on HPC with 2 GPUs

Running LAMMPS interactively

```
[user@mike2 LAMMPS]$ srun -N1 –n32 -p gpu --gres=gpu:2 --time=12:00:00 -A hpc_hpcadmin8 --pty bash
[user@mike2 LAMMPS]$ module purge
[user@mike2 LAMMPS]$ module load lammps/02Aug2023/intel-2021.5.0-cuda-11.6.0-intel-mpi-2021.5.1
[user@mike2 LAMMPS]$ module list

Currently Loaded Modulefiles:
 1) intel/2021.5.0   2) intel-mpi/2021.5.1   3) lammps/02Aug2023/intel-2021.5.0-cuda-11.6.0-intel-mpi-2021.5.1


[user@mike179 LAMMPS]$ time srun --overlap -N1 –n32 lmp_mpi -sf gpu -pk gpu 2 neigh yes newton off -in lj.in > lj.out &

[user@mike179 LAMMPS]$ nvidia-smi –l
```

Running LAMMPS jobs on HPC with GPU

Running LAMMPS jobs using 1 GPU

```
#!/bin/bash
#SBATCH -p gpu
#SBATCH –gres=gpu:1
#SBATCH -N 1
#SBATCH -n 16
#SBATCH -c 1
#SBATCH -t HH:MM:SS
#SBATCH -A hpc_allocation
#SBATCH –J test
#SBATCH -o lammps.%j.out
#SBATCH -e lammps.%j.err
#SBATCH --mail-user=your@email.address

module purge
module load lammps/02Aug2023/intel-2021.5.0-cuda-11.6.0-intel-mpi-2021.5.1

echo $SLURM_JOBID
echo $SLURM_NNODES
echo $SLURM_NTASKS
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

echo $SLURM_SUBMIT_DIR
cd  $SLURM_SUBMIT_DIR

time srun -N1 -n16 lmp_mpi -sf gpu -pk gpu 1 neigh yes newton off -in lj.in > lj.out
```

Running LAMMPS jobs using 2 GPUs

```
#!/bin/bash
#SBATCH -p gpu
#SBATCH –gres=gpu:2
#SBATCH -N 1
#SBATCH -n 32
#SBATCH -c 1
#SBATCH -t HH:MM:SS
#SBATCH -A hpc_allocation
#SBATCH –J test
#SBATCH -o lammps.%j.out
#SBATCH -e lammps.%j.err
#SBATCH --mail-user=your@email.address

module purge
module load lammps/02Aug2023/intel-2021.5.0-cuda-11.6.0-intel-mpi-2021.5.1

echo $SLURM_JOBID
echo $SLURM_NNODES
echo $SLURM_NTASKS
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

echo $SLURM_SUBMIT_DIR
cd  $SLURM_SUBMIT_DIR

time srun -N1 –n32 lmp_mpi -sf gpu -pk gpu 2 neigh yes newton off -in lj.in > lj.out
```

Running AMBER jobs on HPC with 1 GPU

Running AMBER interactively

```
[user@mike2 AMBER]$ srun -N1 -n16 -p gpu --gres=gpu:1 --time=12:00:00 -A hpc_hpcadmin8 --pty bash
[user@mike2 AMBER]$ module purge
[user@mike2 AMBER]$ module load amber/22/intel-2021.5.0-cuda-11.5.0-intel-mpi-2021.5.1
[user@mike2 AMBER]$ module list

Currently Loaded Modulefiles:
  1) intel/2021.5.0   2) intel-mpi/2021.5.1   3) amber/22/intel-2021.5.0-cuda-11.5.0-intel-mpi-2021.5.1

[user@mike2 AMBER]$ export CUDA_VISIBLE_DEVICES="0"

[user@mike2 AMBER]$ time srun --overlap pmemd.cuda –AllowSmallBox -O -i eql.inp -o eql.out -p tip3p_512.prmtop -c tip3p_512.inpcrd -r tip3p_512.rst &

[user@mike2 AMBER]$ nvidia-smi –l
```

DPFP – stands for Double Precision Floating point (64 bits)

SPFP – stands for Single Precision Floating point (32 bits)

Running AMBER jobs on HPC with 2 GPUs

Running AMBER interactively

```
[user@mike2 AMBER]$ srun -N1 –n32 -p gpu --gres=gpu:2 --time=12:00:00 -A hpc_hpcadmin8 --pty bash
[user@mike2 AMBER]$ module purge
[user@mike2 AMBER]$ module load amber/22/intel-2021.5.0-cuda-11.5.0-intel-mpi-2021.5.1
[user@mike2 AMBER]$ module list

Currently Loaded Modulefiles:
  1) intel/2021.5.0   2) intel-mpi/2021.5.1   3) amber/22/intel-2021.5.0-cuda-11.5.0-intel-mpi-2021.5.1

[user@mike2 AMBER]$ export CUDA_VISIBLE_DEVICES="0,1"

[user@mike2 AMBER]$ time srun --overlap pmemd.cuda.MPI –AllowSmallBox -O -i eql.inp -o eql.out -p tip3p_512.prmtop -c tip3p_512.inpcrd -r tip3p_512.rst &

[user@mike2 AMBER]$ nvidia-smi –l
```

DPFP – stands for Double Precision Floating point (64 bits)

SPFP – stands for Single Precision Floating point (32 bits)

Running AMBER jobs on HPC with GPU

Running AMBER jobs using 1 GPU

```
#!/bin/bash
#SBATCH –p gpu
#SBATCH –gres=gpu:1
#SBATCH -N 1
#SBATCH –n 16
#SBATCH -c 1
#SBATCH -t HH:MM:SS
#SBATCH -A loni_allocation
#SBATCH -J test
#SBATCH -o amber.%j.out
#SBATCH -e amber.%j.err
#SBATCH --mail-user=your@email.address

module purge
module load amber/22/intel-2021.5.0-cuda-11.5.0-intel-mpi-2021.5.1

export CUDA_VISIBLE_DEVICES=“0”
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

echo $SLURM_SUBMIT_DIR
cd   $SLURM_SUBMIT_DIR

time srun pmemd.cuda –AllowSmallBox -O -i eql.inp -o eql.out -p tip3p_512.prmtop -c
tip3p_512.inpcrd -r tip3p_512.rst
```

Running AMBER jobs using 2 GPUs

```
#!/bin/bash
#SBATCH –p gpu
#SBATCH –gres=gpu:2
#SBATCH -N 1
#SBATCH –n 32
#SBATCH -c 1
#SBATCH -t HH:MM:SS
#SBATCH -A loni_allocation
#SBATCH -J test
#SBATCH -o amber.%j.out
#SBATCH -e amber.%j.err
#SBATCH --mail-user your@email.address

module purge
module load amber/22/intel-2021.5.0-cuda-11.5.0-intel-mpi-2021.5.1

export CUDA_VISIBLE_DEVICES=“0,1”
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

echo $SLURM_SUBMIT_DIR
cd   $SLURM_SUBMIT_DIR

time srun pmemd.cuda.MPI –AllowSmallBox -O -i eql.inp -o eql.out -p tip3p_512.prmtop -c
tip3p_512.inpcrd -r tip3p_512.rst
```

Running AMBER jobs on HPC with GPU

Running AMBER jobs using SLURM system

```
#!/bin/bash
#SBATCH --partition=gpu
#SBATCH --gres=gpu:2
#SBATCH --nodes=1
#SBATCH --ntasks=32
#SBATCH --cpus-per-task=1
#SBATCH --time=HH:MM:SS
#SBATCH --account=loni_allocation
#SBATCH --job-name=test
#SBATCH --output=amber.%j.out
#SBATCH --error=amber.%j.err
#SBATCH --mail-user=your@email.address

module purge
module load amber/22/intel-2021.5.0-cuda-11.5.0-intel-mpi-2021.5.1

export CUDA_VISIBLE_DEVICES="0,1"
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

echo $SLURM_SUBMIT_DIR
cd   $SLURM_SUBMIT_DIR

time srun pmemd.cuda.MPI –AllowSmallBox -O -i eql.inp  -o eql.out -p tip3p_512.prmtop
-c tip3p_512.inpcrd -r tip3p_512.rst
```

Running AMBER jobs using SLURM system

```
#!/bin/bash
#SBATCH –p gpu
#SBATCH –gres=gpu:2
#SBATCH -N 1
#SBATCH –n 32
#SBATCH -c 1
#SBATCH -t HH:MM:SS
#SBATCH -A loni_allocation
#SBATCH -J test
#SBATCH -o amber.%j.out
#SBATCH -e amber.%j.err
#SBATCH --mail-user=your@email.address

module purge
module load amber/22/intel-2021.5.0-cuda-11.5.0-intel-mpi-2021.5.1

export CUDA_VISIBLE_DEVICES="0,1"
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

echo $SLURM_SUBMIT_DIR
cd   $SLURM_SUBMIT_DIR

time srun pmemd.cuda.MPI –AllowSmallBox -O -i eql.inp     -o eql.out -p tip3p_512.prmtop -c
tip3p_512.inpcrd -r tip3p_512.rst
```

GROMACS Commands Overview

Here's a list of the most common GROMACS commands and their usage:

- **gmx pdb2gmx**: Converts PDB files to GROMACS format, generates topology files, and assigns force fields.
- **gmx editconf**: Defines the simulation box and positions the molecule inside it.
- **gmx solvate**: Solvates the system with water molecules.
- **gmx genion**: Adds ions to neutralize the system.
- **gmx grompp**: Prepares the input for the simulation (creates a .tpr file).
- **gmx mdrun**: Runs the molecular dynamics simulation.
- **gmx energy**: Extracts energy terms from the simulation output.
- **gmx trjconv**: Processes and manipulates trajectory files.

To resume a simulation from a checkpoint file, you use the -cpi flag with the gmx mdrun command to specify the checkpoint file you want to resume from.

gmx mdrun **-deffnm** simulation **-cpi** checkpointfile.cpt

--deffnm specifies the **default file name prefix** for all output files generated by mdrun.

-v is the **verbose mode** flag, which causes GROMACS to print **more detailed output** to the terminal.

***.tpr** The file generated by **grompp** contains all the parameters.

***.gro** The final structure (coordinates) of the system after the simulation.

***.trr** The trajectory file containing positions of atoms over time (optional).

***.edr** The energy file, which contains energy data over the course of the simulation.

***.log** The log file containing detailed information about the simulation progress.

***.cpt** The **checkpoint file** that stores the state of a simulation.

Job state summary for HPC

State	Acronym	Description
PENDING	PD	Job is waiting to be scheduled.
RUNNING	R	Job is actively running.
COMPLETED	CD	Job has finished successfully.
FAILED	F	Job encountered an error.
CANCELLED	CG/CA	Job was canceled.
TIMEOUT	TO	Job exceeded the time limit.

Employing GNUPlot on HPC

```
username@mike2 $ ssh –X mikeXXX
username@mikeXXX $ which gnuplot
username@mikeXXX $ gnuplot
```

```
G N U P L O T
Version 5.2 patchlevel 4    last modified 2018-06-01

Copyright (C) 1986-1993, 1998, 2004, 2007-2018
Thomas Williams, Colin Kelley and many others

gnuplot home:     http://www.gnuplot.info
faq, bugs, etc:   type "help FAQ"
immediate help:   type "help"  (plot window: hit 'h')

Terminal type is now 'qt'

gnuplot>

gnuplot> test

gnuplot> exit
```

plot_results.gp

```
#! /usr/bin/env gnuplot
set title "Temperature vs Time"
set xlabel "Time Step"
set ylabel "Energy (kcal/mol)"
set grid
set key outside

plot [1000:] [200:400] "tip4p_512.out" using 1:2 w l

do for [i=0:*] {
   pause 1
   replot
}
```

http://www.gnuplot.info