Tinker-HP : Readme/Quickstart

October 26, 2017

1 Prerequesites

1.1 Libraries

Tinker-HP requires the MKL library, a FFT library (such as FFTW) and a slightly modified 2DECOMP&FFT library in order to run. The 2DECOMP&FFT library enables parallel 3D FFT computations based on 2d-pencils data distribution (see http://http://www.2decomp.org) based on a sequential implementation of FFTs such as the one provided by the FFTW library.
Besides, Tinker-HP requires a recent enough MPI library supporting MPI 3.x standards such as non blocking collectives to be compiled and to run. The code has been extensively tested with recent intel MPI versions (such as intel MPI 5.1) and better performances have been observed with this family of MPI implementation compared to other ones such as OpenMPI.

1.1.1 Installation of the 2decompFFT library

The sources of the 2decomp&FFT library can be found in the 2decompFFT directory of the distribution. The file that should be modified is "Makefile.inc" in the src directory. The variables that can be modified are "FFT" (default is set to FFT=fftw3_f03 which corresponds to recent versions of the FFTW library) and FFTW_PATH if you are using 2decomp&FFT with FFTW.
Once these variables are set, the compilation can be done by simply running the "make" command in the 2decompFFT directory.

2 Installation

Two Makefiles are available in the source directory of the distribution: "Makefile" which is set to be used with the intel ifort compiler and "Makefile.gfortran" which is set to be used with the gfortran compiler.
In these files, the variables that should be modified are "MKL" which refers to the location of the MKL library, "FFTDIR" which refers to the location of the sequential FFT library (such as FFTW) 2decomp&FFT is compiled with and "FFTDECOMPDIR" which refers to the location of the 2decomp&FFT library. Once these keywords are set, the compilation can be done by simply running the "make" command. The obtained binaries will then be moved to the ../bin/ directory.

3 Executables

After having succesfully compiled the code, four executable files should be present in the bin directory: analyze, dynamic, testgrad and minimize, which are the analogous of the binaries of the Tinker-8.2.1 release and that require similarly a geometry (given by a *.xyz file), a simulation setup (given by a *.key file) and possibly a restart (given by a *.dyn file) for the dynamic program.
All these executables should be launched with the "mpirun -np x" prefix in order to run in parallel with x MPI processes.

analyze The "analyze" executable allows potential energy analysis. Compared to the Tinker-8.2.1 software, the only option compatible with this binary is "e".
For example the command line:
mpirun -np 16 ./analyze dhfr2 e

will give you as an output the potential energy terms of the geometry given by a dhfr2.xyz file and with the simulation setup given by the dhfr2.key file. Furthermore, this computation will run on 16 MPI processes.

dynamic The "dynamic" executable allows to run molecular dynamics simulation. As for Tinker-8.2.1, the command line used to run the MD should give first the number of MD steps to make, then the size of each time step (in femtoseconds), then the time between each writing of geometry (in picoseconds), then the statistical ensemble to sample : 1 is NVE, 2 is NVT, 4 is NPT. For NVT and NPT simulations, this number should be followed by the temperature (in Kelvin) of the simulation, and for NPT simulation by the pressure (in Atmosphere) of the simulation.

For example the command line:

mpirun -np 16 dynamic dhfr2 1000 1 1 1

will give you as an output 1000 MD steps in NVE for the dhfr2 system, with a 1 fs time step and a 1 ps frequency output.

mpirun -np 16 dynamic dhfr2 1000 1 1 2 300

will give you as an output 1000 MD steps in NVT at 300K for the dhfr2 system, with a 1 fs time step and a 1 ps frequency output.

mpirun -np 16 ./dynamic dhfr2 1000 1 1 4 300 1

will give you as an output 1000 MD steps in NPT at 300K and 1atm for the dhfr2 system, with a 1 fs time step and a 1 ps frequency output.

testgrad The "testgrad" program is perfectly equivalent to the one of the Tinker-8.2.1 release: it allows the output of the components of the analytical and/or numerical gradients of the different energy terms.

For example, the command line:

mpirun -np 16 ./testgrad dhfr2 Y Y 0.0001 Y

will give you as an output all the analytical and numerical gradients (computed with an increment of 0.0001 Angstroms for the positions of the atoms) of all the energy terms of the dhfr2 system.

minimize The "minimize" program computes energy minimization starting from a given structure, using a low memory quasi-newton BFGS algorithm as in Tinker-8.2.1. The command line used should give the numerical threshold for the convergence of the algorithm.

For example, the command line:

mpirun -np 16 ./minimize dhfr2 0.1

will compute energy minimization on the dhfr2 structure until the RMS on the gradient is inferior to 0.1. The new geometry will be written at each iteration of the algorithm in the file dhfr2.xyz_2.

4 Keywords

The main keywords of Tinker-8.2.1 are available in Tinker-HP. Let us, review a few of these and some new ones which are specific to Tinker-HP.

keywords specific to the dynamic program

integrators: The integrators available in Tinker-HP are beeman (which is the default one), respa (with a time step for bonded terms which is half the one for non-bonded terms) and verlet. They can be specified by putting the keywords: **integrator beeman**, **integrator respa**, **integrator verlet** in the key file.
thermostats and barostats: The thermostats available in Tinker-HP are Berendsen and Bussi (which is the default one) and the barostat available in Tinker-HP is the Berendsen one. These option can be set by putting the keywords: **thermostat berendsen**, **thermostat bussi** and **barostat berendsen** in the key file.
Tinker-HP deals with restart files for dynamic trajectories the same way as Tinker-8.2.1 does by creating a *.dyn file encompassing current positions, velocities and accelerations of the system.

new keywords specific to Tinker-HP

Some new keywords have been introduced in Tinker-HP. The first one concern the algortihm used to converged the polarization equations. The keyword **polar-alg** has been introduced with the value 1 corresponding to a preconditionned conjugate gradient and the value 2 (which is the default) to a Jacobi/DIIS algorithm.
Unlike Tinker-8.2.1, the neighbor-lists for non bonded interactions are computed every x steps (x=20 being the default) and not adjusted dynamically at each time step. This frequency of update can be modified by using the keywords **nlupdate x**, 20 being the default and 1 corresponding to a neighbor-list update at each time step (which can be useful during equilibration for example).

Other keywords have been introduced to specify parallel options when running Tinker-HP. The current distributed release is only available with the PME algorithm which involves direct and reciprocal space computations in the electrostatic and polarization interactions. As the reciprocal space interactions are known to have a less efficient parallel scaling (because of FFTs) it is possible to specify a lower number of MPI processes that will be dedicated to these computations for both the computation of electrostatic and polarization interactions. This can be done by using the keyword **pme-procs x**, corresponding to x MPI processes dedicated to reciprocal space computations.

A good starting point, when a large number of cores is used, is usually to dedicate about $\frac{1}{4}$ of the total cores to reciprocal space computations but this parameter depends greatly on the machine used and on the setup of your simulation so it should be adjusted by hand by comparing the timings obtained with different values for **pme-procs**. During a dynamic, detailed timings are written when the **verbose** keyword is in the *.key file. In the future, this parameter will be adapted heuristically by the program as it is done in popular MD packages.

As explained in the beginning of the document, Tinker-HP distributes the data on a grid used to run 3D FFT in 2d pencils which can be associated to a 2d processor grid as explained in : http://www.2decomp.org/decomp.html. By default, the library looks for an optimal 2d processor grid given the number of cores available but this decomposition can be imposed by the user by setting the keyword: **decompfft-grid n1 n2**, corresponding to a n1×n2 processor grid.

Note that the user does not have to specify that he wants neighbor-lists to be used as it is the only available option in Tinker-HP.

5 Examples

4 examples of systems with associated *.key files are given in the distribution: ubiquitin2, dhfr2, puddle and pond. The sizes of these systems are respectively: 9737, 23558, 96000 and 288000 atoms, making them good various benchmarks for the program.