Using Multiple Walkers in the built-in Metadynamics module

Multiple Walkers.

The usage of **multiple walkers** can be extremely useful for speeding-up the filling of the Free Energy Surfaces.
The implementation available in CP2K relies on communication via file.

The idea is to allocate a large amount of processor split in several groups (via FARMING) and each group will run an independent metadynamic. Every CT metadynamic step all the walker (independently) will try to read information on the status of the other **walkers**. If possible, they will acquire the hills, which have been spawned by the other **walkers**, merge them with their own hills and continuing the exploration of the free energy surface.

Needed Information.

Every **walkers** need to know its personal ID, the number of total **walkers** and the files where the **walkers** will dump information on the "local" metadynamic run.

Input file structure.

```
@SET ROOT /tmp/teo/
@INCLUDE 'my_force_eval.inc'&MOTION
 &MD
  ENSEMBLE NVT
  STEPS 5000
  TIMESTEP 0.5
  TEMPERATURE 300.0
  &THERMOSTAT
   REGION MASSIVE
   &NOSE
    LENGTH 3
    YOSHIDA 3
    TIMECON 1000.
    MTS 2
   &END NOSE
  &END
 &END MD
 &PRINT
  &RESTART
   &EACH
    MD 100
   &END
  &END
  &TRAJECTORY
   &EACH
    MD 20
   &END
  &END
  &VELOCITIES OFF
  &END
 &END
```

```
 &FREE_ENERGY
  &METADYN
   NT_HILLS 40
   WW 1.0E-3
   &METAVAR
    SCALE 0.02
    COLVAR 1
   &END METAVAR
   &MULTIPLE_WALKERS
    NUMBER_OF_WALKERS <NW>
    WALKER_ID <ID>
    &WALKERS_FILE_NAME
@INCLUDE ../map_walker.inc
    &END
   &END
   &PRINT
    &COLVAR
     COMMON_ITERATION_LEVELS 4
     &EACH
      METADYNAMICS 1
     &END
    &END
    &HILLS OFF
    &END
   &END
  &END METADYN
 &END
&END MOTION
```

In the above input template the metadynamic part of the is in red. Yellow-highlighted is the part specific to the **multiple walkers**.
Normally, you could substitute the <NW> with the total number of **walkers**, <ID> with the ID counter of the local walker and add
a file with all the file names (together with the proper path) used by the **walkers** to dump hills information (which will be read by the rest of the **walkers** pool).
Using this script will allow you to setup automatically all those variables.
Follow these instructions:

1. Create a working directory for your project (~>mkdir my_project)
2. Create a template file: template.MW (the one above is a good starting point.. don't forget to add your force_eval section and to specify properly the COLVAR and the METAVAR in the &SUBSYS section and in the &METADYN section). The template file should be called template.MW. Leave the MULTIPLE_WALKERS section according the setup above. Don't specify <NW>, <ID> or the **walkers** file name in the input.. the script will take care of them directly.
3. run the script : ./get_walk.csh 32 ; where 32 is the number of walker that you want
4. The script will generate 32 directories (containing the properly setup input file for each walker) and the input file for the farming..
5. Run the farming input file..

Note

It is important when running **multiple walkers** to start each walker from different point in the phase space. To this aim I would recommend to run first an equilibration MD and save the restart file (in a number equal to the **walkers**) every a certain amount of time. In case several basins are already known, the **walkers** should ideally start also from these points.

General info.

- Communication via file may not be an optimal choice (due to the high IO), but allows (up to 256 tested **walkers**) for an optimal scalability and was chosen mostly for its simplicity of implementation. According on the IO capabilities of the machine even smaller number of **walkers** (128 or 64) may overload the IO system. So care should be taken when deciding for the right number of **walkers**.
- The file layer communication allows for an extremely independent execution of the runs. This means that if any of the walker should fail during **metadynamics** the other **walkers** would continue anyway.
- Restarting **metadynamics** with **multiple walkers** is as easy as restarting a standard **metadynamics**. In fact, when additional hills coming from other **walkers** are read, they are merged with the local hills and they are dumped in the restart of each walker. It is therefore enough to restart **metadynamics** in the usual way. An additional keyword (RESTART_WALKERS) can disable the restart of the **walkers** information in case of changing the number of **walkers** or failures of 1 **walkers** during the run.