

How to run DPD simulations on helvetios at EPFL

We have computer time on the *helvetios.hpc.epfl.ch* HPC machine for the course, so you can run longer/larger simulations than are feasible on your laptop. If you have access to the cluster through your lab, you don't have to use this procedure, you can just run the code yourself.

To run the DPD code on helvetios you have to copy the executable **hdpd** to helvetios (or compile it on there yourself from the source), and your input files, create a job script, and submit the simulations using the script file. The default maximum length is 3 days. If you want to run up to 10 days, you need to add the quality of service (qos):

```
#SBATCH --account=bio-692
#SBATCH --qos bio692
```

You can estimate how long a job will take (this is for my desktop, and I haven't timed this on helvetios, and it depends on the processor speed, etc) by calculating:

$\# \text{ of cpu-days} = \# \text{ of beads} * \# \text{ of time-steps} / 5 * 10^{**10}$

If your box is 20^{**3} , with a density of 3 beads/unit volume, and you run 100,000 time-steps, it should take

$3*8000*100000 / 5*10^{**10} = 0.048 = 1.15 \text{ hours.}$

I have put two typical job scripts up on the moodle home page.

Script name	-qos	—ntasks	Number of simulations	Input file names
job-1	normal	1	1	dmpci.001

Here is the script “job-normal-1” that runs one job in the normal queue for 24 hours:

```
#!/bin/bash

#SBATCH --account=bio-692
#SBATCH --job-name="my-job-name"
#SBATCH --qos bio692
#SBATCH --nodes 1
#SBATCH --ntasks-per-core 1
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu 1024
#SBATCH --time 24:00:00
#SBATCH --output out-%J.txt
#SBATCH --error err-%J.txt

../hdpd 001&
wait
```

The lines you have to modify to run different numbers of simulations/different times are in red. You should set *my-job-name* to a short string so you can identify your different jobs.

The script assumes that the DPD executable file is called **hdpd** and is in the directory above where the script and the input files are located.

So, for example, if you want to run 3 jobs (called dmpci.001, dmpci.002, dmpci.003) for 10 days, you would use the following script:

```
#!/bin/bash

#SBATCH --account=bio-692
```

```
#SBATCH --job-name="my-job-name"
#SBATCH --qos bio692
#SBATCH --nodes 1
#SBATCH --ntasks-per-core 1
#SBATCH --ntasks=3
#SBATCH --mem-per-cpu 1024
#SBATCH --time 10-00:00:00
#SBATCH --output out-%J.txt
#SBATCH --error err-%J.txt
```

```
../hdpd 001&
../hdpd 002&
../hdpd 003&
wait
```

The number in the line “`--ntasks=3`” **must** match the number of runs started at the end of the script, and there must be ampersands (&) at the end of each executable line.

To summarise, in order to run jobs on helvetios, you have to do the following:

- 1) Create your run script and input file on your local machine
- 2) Put the files onto helvetios in the right directories
- 3) Submit the script and track its progress

1 Creating your files

Copy and edit one of the provided scripts and create your ddp input files. I recommend trying the “`job-normal-1`” script with the single input file “`dmpci.001`” also on moodle. This should only take a few minutes to run, and you can check it works. Then you can copy and edit it for your own runs.

2) Put the files onto helvetios

Assuming the script file and the input file(s) are in the current directory on your local machine, and you open a terminal window in that directory, execute the following commands (if you’re off campus, you’ll need to VPN onto campus to be allowed onto helvetios). You must replace “`username`” by your gaspar username:

```
> sftp username@helvetios.hpc.epfl.ch      (it will prompt for your
password)
```

```

> put hpd                (or your executable, only do this once!)
> put job1                (or whatever your script is called)
> put dmpci.001           (or whatever your input is called)
> quit

```

3) Now login to helvetios and submit the script to run the jobs

```

> ssh username@helvetios.hpc.epfl.ch      (it will prompt for
your password)

> mkdir test                            (create a sub-directory called test,
                                         unless it already exists from a
                                         previous run)

> mv job1 test                          (move the script into the sub-dir)

> mv dmpci.001 test                     (move input files into the sub-dir)

> cd test                               (change directory to test)

> sbatch job1                           (submit the script)

> squeue -u username                    (replace username by your gaspar
username)

> exit

```

The “squeue” command shows you a list of your running jobs. It can be used to track how far the job has run. Its output looks like this:

```

[shillcoc@fidis llps-membrane]$ jobs

```

	JOBID	PARTITION	NAME	USER	ST	TIME	NODES	N
1	12264333	serial	llps2	shillcoc	R	47:46	1	f

I recommend using a different sub-directory for each problem or project to keep the files separate. Given that dpd can produce many output files, especially snapshots, it gets crowded quickly.

You can track your jobs by logging into helvetios and running the `squeue` command again and it will show you how long it has been running, e.g.

4) Get the files back from helvetios

To retrieve files you use the `sftp` command “`get`”. It’s a good idea to run related jobs in a single directory because then you could get them all back using simply “`get *.*`”. But assuming the input file was `dmpci.001`, you get the results using these commands:

```
> sftp username@helvetios.hpc.epfl.ch      (it will prompt for your
password)

> cd test                                (assuming the run was in ~/test)

> get *.001                             (or whatever your runs were called)

> get dmpccs.001.*                       ( not all output files end in 001)

> get dmpcrs.001.*                       ( restart files are large, so only do
this if you really want them)

> quit
```

NB You can also compress the files using `zip *.*` if you have lots of snapshots, for example, then you only have to retrieve the single zip file.