







Running GROMACS on LUMI workshop 24.-25.01.2024 Q&A




This is the collaborative "notebook" for the GROMACS on LUMI online workshop organized by CSC and BioExcel on 24-25 January 2024.

 The **workshop schedule** can be found here.

 This is also the place to ask questions about the content!

 For exercise instructions, see this page! (<https://hackmd.io/@GROMACS-LUMI/Hkf-BO4rp>)

 **Hint:** HackMD is great for sharing information in this kind of courses, as the code formatting is nice & easy with Markdown (<https://www.markdownguide.org/basic-syntax/>)! Just add backticks (` `) for the code blocks . Otherwise, it's like Google docs as it allows simultaneous editing. There's a section for practice at the bottom 

- Running GROMACS on LUMI workshop 24.-25.01.2024 Q&A
 -  How we work
 - Overall setup
 - Other Useful links
 - General instructions for LUMI
 - Code of conduct
 - Lecture Presentations
 -  Agenda
 - Day 1: Wednesday 24.01.2024 (times CET)
 - Day 2: Thursday 25.01.2024 (times CET)
 -  Q&A – type your questions here!
 - Feedback
 - Icebreaker

- 🐉 HackMD practice area

How we work

Overall setup

1. Zoom (<https://cscfi.zoom.us/j/66544791501?pwd=R1FycUJQOWNYR0YrRFNMSTViMnd2UT09>) for lectures and general hands-on support
2. **(This)** (<https://hackmd.io/l3V4xWFCSAiPwnMOFJSVPA>) collaborative HackMD page for info & Q&A
3. Another HackMD (<https://hackmd.io/@GROMACS-LUMI/Hkf-BO4rp>) for hands-on instructions

Other Useful links

- CSC GROMACS on LUMI Documentation (<https://docs.csc.fi/apps/gromacs/#example-batch-script-for-lumi-single-gcd>)
- LUMI general Documentation (<https://docs.lumi-supercomputer.eu/>)
- Contacts for any follow-up questions: GROMACS forum (<https://gromacs.bioexcel.eu/c/gromacs-user-forum/5>)
- HackMD cheat sheet (<https://hackmd.io/@hackmd/markdown-cheat-sheet>) (you can also click the question mark at the top)

General instructions for LUMI

- You can login to LUMI using SSH: `ssh <username>@lumi.csc.fi` **or** using the LUMI web interface (<https://www.lumi.csc.fi>) (open a Login node shell). You must have set up and registered your SSH key pair (<https://docs.lumi-supercomputer.eu/firststeps/SSH-keys/>) prior to this.
- All simulations should be launched from the scratch folder of the training project (/ scratch/project_465000934) and in your own respective folders. Don't launch jobs from your \$HOME directory.
- We recommend any of the following text editors: *vi*, *vim*, *emacs*, *nano* for editing files. Note that *gedit* is not available on LUMI.
- Please double check your submission script (.sh file) for the #SBATCH details before launching simulations. Ensure that all the details are entered correctly. Common Slurm options and error messages are detailed in the LUMI documentation (<https://docs.lumi-supercomputer.eu/runjobs/scheduled-jobs/batch-job/>).

- Please note that Exercise 1.1 and 1.3 are for demonstration purposes only. One of the main motivation for running GROMACS on LUMI-G is to take advantage of the enhanced performance provided by the GPUs. Utilizing only the CPU cores of a GPU node for production simulations is wasteful use of resources and not allowed.

Code of conduct

- ▶ Details

Lecture Presentations

You can find all the lecture slides for the workshop **here** (<https://zenodo.org/records/10556522>)



Agenda

Day 1: Wednesday 24.01.2024 (times CET)

Time	Content
9:30	Workshop Introduction (communication channels, icebreaker with HackMD, code of conduct)
9:40	LUMI Architecture (https://zenodo.org/records/10683366/files/lumi-arch.pdf) + Q/A session
10:10	Brief Introduction to GROMACS (https://zenodo.org/records/10683366/files/Short_Intro_GROMACS_LUMI.pdf) + Q/A session
10:25	<i>Break (15 min)</i>
10:40	GROMACS parallelization / heterogeneous/GPU algorithms (https://zenodo.org/records/10683366/files/LUMI-WS_Szilard-Pall_GROMACS-parallelization.pdf) + Q/A session
11:30	<i>Lunch break (75 min)</i>
12:45	AMD GPU support in GROMACS (https://zenodo.org/records/10683366/files/AMD-GPU-support-in-GROMACS.pdf) + Q/A session
13:15	How to run GROMACS on LUMI (Slurm; batch script option) (https://zenodo.org/records/10683366/files/lumi-batch.pdf) + Q/A session
13:30	<i>Break (15 min)</i>
13:45	Introduction to Hands-on session
14:05	Hands-on session instructions (https://zenodo.org/records/10683366/files/Running_molecular_dynamics_simulations_with_GROMACS_on_LUMI-HackMD.pdf)
15:20	Finish day 1

Day 2: Thursday 25.01.2024 (times CET)

Time	Content
9:30	Assessing and tuning performance of GROMACS simulations (https://zenodo.org/records/10683366/files/) LUMI-WS_Szilard-Pall_Assessing_and_tuning_GROMACS_performance_on_heterogeneous_systems.p + Q/A session
10:00	<i>Break (15 min)</i>
10:15	Hands-on: GPU accelerated simulations instructions (https://zenodo.org/records/10683366/files/Running_molecular_dynamics_simulations_with_GROMACS_on_LUMI-HackMD.pdf)
11:30	<i>Lunch break (75 min)</i>
12:45	Hands-on: Scaling GROMACS across multiple GPUs instructions (https://zenodo.org/records/10683366/files/Running_molecular_dynamics_simulations_with_GROMACS_on_LUMI-HackMD.pdf)
14:00	<i>Break (15 min)</i>
14:15	Hands-on: Ensemble parallelization across multiple GPUs instructions (https://zenodo.org/records/10683366/files/Running_molecular_dynamics_simulations_with_GROMACS_on_LUMI-HackMD.pdf)
15:30	Finish day 2

! ? Q&A – type your questions here!

Q1: How to get access to LUMI-D?

- A: LUMI-D large memory nodes can be accessed by using `--partition=largemem` in your batch job. LUMI-D visualization nodes can be accessed through the LUMI web interface accelerated visualization app.

Q2: Is there a difference between CPU and socket?

- A: A single LUMI-C node has 2 sockets or processors. Each socket in turn has 64 CPU cores.

Q3: Do we get the slides?

- A: Yes. The slides will be updated/linked to the agenda.

✓ **Q4: Is it possible to run GPU jobs across multiple nodes ?**

- A: Yes. It largely depends upon the size of the system that you're planning to simulate. Please check this link for more details: <https://docs.csc.fi/apps/gromacs/#example-batch-script-for-lumi-single-gcd> (<https://docs.csc.fi/apps/gromacs/#example-batch-script-for-lumi-single-gcd>). Another way would be to run multiple independent simulations at the same time using e.g. `-multidir` (<https://docs.csc.fi/support/tutorials/gromacs-throughput/> (<https://docs.csc.fi/support/tutorials/gromacs-throughput/>))

✓ **Q5: How to estimate the scaling limit for your system (i.e., if it's feasible to run on multiple nodes)?**

- A: You have to benchmark your system to see how well it scales across multiple GCDs or nodes. This will be dealt in detail during the hands-on exercise session.

✓ **Q6: Is it possible to also run jobs on e.g. LUMI-G and LUMI-C in parallel, with communication via high-speed interconnect ?**

- A: In principle, yes. But you would require to submit jobs separately on LUMI-G and LUMI-C using separate submit scripts. This could, however, be orchestrated/automated using some external workflow manager.

✓ **Q7: Would the uncoupled ensembles have the advantage of being unbiased? how bias would account in the ensemble paralelization choice?**

- A: The bias itself is not necessarily connected to the coupling of ensembles. It highly depends on the method used.

✓ **Q8: What is the frequency of communication between domains and where it is done?**

- A: All forces are computed withing the inner-most loop (every step), this includes exchange of non-local coordinates (at the beginning of the step) and non-local forces (at the end of the step) in the halo region.

✓ **Q9: `nstlist=20-40` was mentioned in previous presentation to work well with GPU offloading. With multi-GPU runs the `nstlist` value can be automatically increased to e.g. 100 to increase performance. The GROMACS documentation recommends in some cases to increase this automatic value manually even more to `nstlist=200-300`. Is there an upper limit for `nstlist` and how to know which value to choose?**

- A: Technically, no upper limit. In practice: higher `nstlist` increases Verlet buffer size, we decrease the workload outside the "blue" MD loop (see slides), but we increase the work in the "blue" loop, and there is a specific crossover point where the sum of these effects is minimal.

✓ **Q10: (related to Q9) What should we consider specifically to choose the correct or most efficient `nstlist` value? Is it just a trial-and-error approach? Just testing different values and compare the performance?**

- A: Benchmarking the performance with different options is indeed a good approach.

✓ **Q11: Are there any specific challenges of installing GROMACS to be able to use AMD GPUs? For example, to make sure the correct GCD-CCD mapping is used (in case the mapping is necessary at that stage).**

- A: The discussion at the moment is about how to ask the queuing system (and tell `mdrun` at the command line) how to map/use the hardware when launching the job. Installation is a separate topic. On LUMI GROMACS is preinstalled.
 - Overall, CPU-GPU mapping is decided when running code, not when compiling.

✓ **Q12: Would it be possible to run with 1 full node + 1(or 2) GPUs? Or if multiple GPU nodes are used, do they need to be always full nodes?**

- A: Ideally you would be required to use a full node, as this provides optimal performance. Requesting full nodes is also a requirement to be able to set CPU-GPU binding properly (exclusive control of resources). Also remember that as the layer of hierarchy increases (core > CCD/GCD > nodes), the data transfer delay also increases.

✓ **Q13: Is the energy minimization supported with GPUs?**

- A: Energy minimization is usually so fast that GPU acceleration is not necessary.
- So do we always have to run the energy minimization using LUMI-C partition only and the next steps using LUMI-G?
- A: It depends :) If you have just one simulation, and you expect the energy minimization to be quick, it does not make sense to queue for a separate (very short) batch job in a LUMI-C partition. If you have 1000 independent systems, it's better to run `grompp`, energy minimization etc. for all of them in a separate CPU-only job *first* and then use these as input to the MD steps. If you reserve a full

GPU-node for energy minimization, you will burn resources all the time - you can try to compare that to the required additional steps/effort to use a separate CPU-only step (on LUMI-C). There is a tipping point somewhere.

✓ **Q14: From my limited experience compiling GROMACS it seems to me GPU support is not compatible with double precision calculations. Is it correct? Is there any plan to extend GPU calculations to double precision in future versions?**

- A: You are right, double-precision is not supported on GPUs. There are, at the moment, no planned work on that.
 - You can keep an eye on GROMACS Announcements (<https://gromacs.bioexcel.eu/c/gromacs-announcements/7>) forum. We are planning to run a user survey this spring/summer, which will include a question about which features you (users) want.

✓ **Q15: Can you explain what "ROCR_VISIBLE_DEVICES" does again?**

- A: It is an environment variable that maps each task (SLURM_LOCALID) to a particular GCD on the node to ensure isolation of tasks. The mapping of specific cores to each task (associated with a specific GCD) is then done using the appropriately reordered list of hexadecimal masks.

✓ **Q16: What distinguishes #SBATCH -gres= from #SBATCH gpu-per-node?**

- A: Whether gres or the other flags are used to request GPUs depends on how Slurm has been set up on the particular cluster. On LUMI, you should use `-gpus-per-node` to set the number of GPUs (or GCDs, actually) per node, while e.g. on CSC supercomputer Puhti and Mahti you would use `-gres=gpu:...` to set the number of gpus per node.

✓ **Q17: Would assigning ntask-per-node > 16 have an impact on efficiency?**

- A: Yes, when running ensemble simulations it might make sense to run several simulations per GPU when running an ensemble (Exercises 4.x tomorrow). More advanced technique: with separate PME ranks, it might make sense to share the same GPU between a PME rank and a PP rank (from the same or another simulation) (Bonus task in Exercise 3.3).

✓ **Q18: How to get production access to LUMI?**

- A: LUMI is a supercomputer jointly owned by EU and a consortium of 10 countries.

The consortium countries can decide on giving resources from their own share (which is proportional to their investment on the hardware) and EU (via EuroHPC JU) gives access to the 50% of the capacity managed by them. The consortium is described in here: <https://www.lumi-supercomputer.eu/get-started/> (<https://www.lumi-supercomputer.eu/get-started/>)

- The EuroHPC JU resources can be used via 4 different modes: Benchmark, Development, Regular and Extreme scale access. To get started, apply for one of the first two (decision should come within a month). The latter calls are announced 2-3 times a year and applications go through scientific and technical evaluation. Use the preparatory access projects to create data to demonstrate that your code performs well. The calls can be accessed from here (also for other systems than LUMI): <https://prace-ri.eu/hpc-access/eurohpc-access/> (<https://prace-ri.eu/hpc-access/eurohpc-access/>)

✓ **Q19: What would be considered good enough scaling?**

- A: Depends on what kind of scaling one is considering. Strong scaling comes with a trade-off, namely parallel efficiency. A large system is typically able to utilize a large amount of resources more efficiently = better parallel efficiency, while a smaller system might not typically be able to use as many cores/GPUs. In the end, you need to decide what is acceptable to you, but also it is important to consider what is efficient use of resources. I.e. it does not make sense to double the amount of resources if the speed up you get is only, say, 1.2 times. A rule of thumb at CSC what is acceptable scaling is that your job should become at least 1.5 times faster when you double the amount of resources.

✓ **Q20: What is "Wait GPU state copy" exactly?**

- A: This is related to the time the CPU has to wait for the GPU to do work.

✓ **Q21: I wondered whether, due to the LUMI arch, one can always use `-gpus-per-node=2` when running gromacs?**

- A: See Q12 :) It is possible to use less than a full node, but this does not allow configuring CPU-GPU affinity in the most efficient way.

✓ **Q22: What if I have 2x22 cores in HPC, then how many `cpu-per-task` can I assign per node?**

- A: $n_{\text{task-per-node}} * \text{cpu-per-task}$ equals the total number of CPU cores in a single node. So if the CPU has 22 cores each and 2 CPUs per node, then ideally

you can run 44 tasks on a single node. However, if your CPU cores support two hardware threads, you can launch twice the amount than you have CPU cores. So in this case $n_{\text{tasks-per-node}} * \text{cpus-per-task}$ could be equal to 88. This is known as simultaneous multi-threading (SMT).

✓ **Q23: So, either use 8 gpus/node or if you have small systems then run multiple small simulations with -multidir?**

- A: Yes. This is recommended practice. It is possible to use less than full node also, but then you cannot do the CPU-GPU binding and might get suboptimal performance.

✓ **Q24: everytime I increase the number of openmpthreads the simulation time keeps on increasing, so I want to understand how ntask-per-node and cpu-per-task correlate?**

- A: Are you referring to total simulation time (it is, in our scripts, always limited to approx. 1 minute) or to the "Core t" reported at the end of the log file? The latter measure CPU resources consumed and is, approximately, $1/2 * (1 \text{ minute}) * (\# \text{ CPU cores})$ – it is expected that if you use more CPUs, you consume more core-time. But the simulation performance, ns/day, should increase.

✓ **Q25: Are we billed based on how much time we ask or how much time we actually use?**

- A: The billing is dependent on the consumption of the resources i.e. it is based on the time your application is running (and blocking the resources from other users).

✓ **Q26: Is there a "best practice" number of steps in scaling test runs? i.e., is there some delay before the dynamic load balancing for example starts to act properly?**

- A: There are several load-balancers and hardware behaviors (clock ramp-up, thermal throttling). A rule of thumb: a few minutes. Exercises are limited to one minute to conserve resources for the workshop. If the simulation is very large and very slow, then at least a few 10'000 of steps, but otherwise focus on run-time.

✓ **Q27: The documentation says that -tunepme is incomaptible with some mdrun options. Could you explain what these incompatibilities mean and how we can spot them? In other words, what happens if I use -tunepme but it is incompatible**

with the simulation system?

- A: Simple answer: mdrun will tell you if `-tunepme` cannot be used. The only major feature that disables PME tuning is PME decomposition (multiple GPUs for PME).

✓ **Q28: I am confused with the terms used in the gromacs output: Using __ MPI process and Using __ OpenMP threads per MPI process. if we talk about running simulations only on lumi CPU where each node has 56 cores (if I'm correct) then what would be the numbers appear on those two dashes to run the simulation in a efficient way without having any problems of dynamic load imbalancing?**

- A: GPU nodes have 56 CPU cores available, LUMI-C nodes have 128 cores. But regardless, you should ideally try different combinations and see how they affect the performance. That is, try different numbers for `--ntasks-per-node` and `--cpus-per-task` (and `OMP_NUM_THREADS`). The total number (`ntasks-per-node*cpus-per-task`) should add up to the number of cores on the node (or twice the amount if using simultaneous multi-threading, which requires you to add option `--hint=multithread`).

✓ **Q29: Does it take time to tune the PME in the beginning and the counters should be reset later because of that?**

- A: Yes, there can be some startup overhead and performance instability (e.g. related to tuning PME or load balancing) at the beginning of the run. It is a good idea to reset the counters using `-resetstep` or `-resethway` to get more reliable performance output.

✓ **Q30: How should we change the nstlist frequency?**

- A: Using the `-nstlist` option. Increasing it can have a positive impact on performance, especially for multi-GPU runs.

✓ **Q31: I tried setting the nstlist = 200 with PME tuning on. I got following error: Fatal error: PME tuning was still active when attempting to reset mdrun counters at step 1901. Try resetting counters later in the run, e.g. with gmx mdrun -resetstep.**

- A: `-resetstep` / `-resethway` is resetting our internal performance counters after the first half of the simulation; it is a useful option to avoid the first few steps skewing the reported performance. But if this coincides with the time PME tuning is being performed, bad things happen. You can either try running longer (2-3

minutes) so that tuning happens in the first half of the run (change `-maxh`) or set `-resetstep` (instead of `-rethway`) such that it happens after the tuning.

✓ **Q32: If I have already ran simulations, would it be safe to increase the `nstlist` for continuation without affecting results? Basically only the performance is changed?**

- A: Yes, accuracy will not be affected.

✓ **Q33: I tried to activate `-tunepme` and I got this error. What can the reason be "Fatal error: PME tuning was still active when attempting to reset mdrun counters at step 451. Try resetting counters later in the run, e.g. with `gmx mdrun -resetstep`. "?**

- A: See answer to Q31 for solution.

✓ **Q34: In exercise 3.2 bonus 1, we are asked to set `-gpus-per-node=2` and `-ntasks-per-node=3`, but then I get this error:**

Inconsistency in user input:

There were 3 GPU tasks found on node `nid005037`, but 2 GPUs were available. If the GPUs are equivalent, then it is usually best to have a number of tasks that is a multiple of the number of GPUs. You should reconsider your GPU task assignment, number of ranks, or your use of the `-nb`, `-pme`, and `-npme` options perhaps after measuring the performance you can get.

- A: Do you mean exercise 3.3?
- yes, sorry
- Did you remember to perform the CPU-GPU binding by sourcing `lumi-affinity.sh` script and using `--cpu-bind=${CPU_BIND}` and `./select_gpu ?`
- oops, for some reason I have not included it. I will fix and re-run
- Good! In this case we need to make sure that the right GPU devices are exposed to the tasks. The script will take care of this, in particular it will make sure one GCD is associated with two tasks (PP+PME) while the other is associated with the remaining PP task.
- much worse performance ns/day 0.117
- This is the jobscript:

```

#SBATCH --nodes=1                    # we run on 1 node
#SBATCH --gpus-per-node=2            # fill in number of GPU devices
#SBATCH --ntasks-per-node=3         # fill in number of MPI ranks

module use /appl/local/csc/modulefiles
module load gromacs/2023.3-gpu

source ${GMXBIN}/lumi-affinity.sh

export OMP_NUM_THREADS=7

export MPICH_GPU_SUPPORT_ENABLED=1
export GMX_ENABLE_DIRECT_GPU_COMM=1
export GMX_FORCE_GPU_AWARE_MPI=1

srun --cpu-bind=${CPU_BIND} ./select_gpu \
    gmx_mpi mdrun -npme 1 \
        -nb gpu -pme gpu -bonded gpu -update gpu \
        -g ex3.1_${SLURM_NTASKS}x${OMP_NUM_THREADS}_jID${SLURM_
        -nsteps -1 -maxh 0.017 -rethway -notunepme

```

- strange, I do not see any obvious issue in the script. Reference result and batch script can be found here (<https://github.com/Lumi-supercomputer/gromacs-on-lumi-workshop/tree/main/Exercise-3.3/STMV/bonus>). If your script looks identical, then I'd suggest you just try running it again.

✓ **Q35: I get error when trying to assign 3 tasks on 2 GPUs for bonus in ex3.2. MPICH ERROR - Abort(1) (rank 0 in comm 0): application called MPI_Abort(MPI_COMM_WORLD, 1) - process 0?**

- A: Does the answer above (Q34) help?
- B: yes, it is solved now. I thought it is part of ex3.2

✓ **Q36: Was it more efficient to run 9 tasks with 8 GPUs? Or is the difference compared to 8 tasks/8 GPUs really minimal?**

- A: Running 9 tasks on 8 GPUs contributes to an imbalance: one GPU would be running 2 tasks while the remaining 7 would be running 1 task each. It would be ideal to run tasks in multiples of the number of GPUs requested. Although the performance would largely depend on the system size. For smaller systems (around 100,000 atoms), you can run more than 1 task per GPU but for larger systems (like STMV) it would be ideal to set 1 task per GPU (or maybe even 1 task

per 2 GPUs).

- A: When I run this configuration the performance drops from ~70 ns/day to about 55 ns/day.

✓ **Q37: In exercise 4.1 the suggested number for OMP_NUM_THREADS are 7/(ntasks-per-node/8). What does this mean? In the previous exercises the number was always 7.**

- A: Since you need to run (at least) as many tasks as you have ensemble members, you will need to adjust your OMP_NUM_THREADS based on the number of tasks (ensemble members). So e.g. for 16 ensemble members, you should have 16 tasks per node, and as there are 56 cores available, you can use at most $OMP_NUM_THREADS=3$, i.e. $7/(16/8)=3.5$ (and round down to 3).

✓ **Q38: Sorry, I am lost. How many MPI ranks should we use in exercise 4.1?**

- A: The same amount as ensemble members (num_multi). So first 8, then 16, 24, 32. And remember to adapt OMP_NUM_THREADS accordingly based on the total number of CPU cores per node (56).

✓ **Q39: I am getting this error To run mdrun in multi-simulation mode, more than one actual simulation is required. The single simulation case is not supported. How can I solve it?**

- A: In Ex. 4.x, you are using `-multidir` mode which is designed for running ensembles. If you want a single simulation, don't use `-multidir`. You should also make sure that `ntasks` is \geq number of ensemble members (number of directories).
- B: I am not sure where I am making mistake. I am in the main directory of aquaporin and the ntask is 7. with 7 threads and 7 ensembles. I get the same error when trying 32 tasks and 1 thread and 32 ensembles.
- Have you changed the ?? after the `-multidir` flag?
- No, that should be it. Thanks.

✓ **Q40: How do I get the aggregate performance?**

- A: Try `grep Performance */ex4.1_${SLURM_NNODES}N_multi${num_multi}_jID${SLURM_JOB_ID}.log` and calculate the sum
- It's the sum of all the average performances of the ensembles.
 - `awk` is also a handy command. You can use it to sum up all performance, for

```
example: grep Perf */ex4.1_${SLURM_NNODES}N_multi${num_multi}_jID$
{SLURM_JOB_ID}.log | awk '{ sum += $2; n++ } END { if (n > 0) print
sum ; }'
```

- Or `grep Perf */ex4.1_${SLURM_NNODES}N_multi${num_multi}_jID$ {SLURM_JOB_ID}.log | awk '{ sum += $2; n++ } END { if (n > 0) print sum / n ; }'` to get the average

☑ **Q41: Sorry, I am completely lost. This is my jobscript:**

```
#SBATCH --nodes=1                # we run on 1 node
#SBATCH --gpus-per-node=8        # the number of GPU devices per node
#SBATCH --ntasks-per-node=8

module use /appl/local/csc/modulefiles
module load gromacs/2023.3-gpu
source ${GMXBIN}/lumi-affinity.sh #

export OMP_NUM_THREADS=7        # fill in the number of threads (7/(ntasks
num_multi=8                     # change ensemble size

srun --cpu-bind=${CPU_BIND} ./select_gpu \
    gmx_mpi mdrun -multidir sim_{01..??} \
    -nb gpu -pme gpu -bonded gpu -update gpu \
    -g ex4.1_${SLURM_NNODES}N_multi${num_multi}_jID${SLURM_JOB_ID} \
    -nsteps -1 -maxh 0.017 -resetway -notunepme
```

I receive the following error immediately after submission:

```
Feature not implemented:
```

```
To run mdrun in multi-simulation mode, more then one actual simulation is
required. The single simulation case is not supported.
```

```
For more information and tips for troubleshooting, please check the GROMACS
website at http://www.gromacs.org/Documentation/Errors
```

```
-----
MPICH ERROR [Rank 5] [job id 5896229.0] [Thu Jan 25 15:51:05 2024] [nid0050:
```

```
aborting job:
```

```
application called MPI_Abort(MPI_COMM_WORLD, 1) - process 5
srun: error: nid005037: tasks 0-7: Exited with exit code 255
srun: launch/slurm: _step_signal: Terminating StepId=5896229.0
```

- A: First, you need to change ?? to the same value as num_multi after the -

multidir flag

- Thanks, it seems to be running now

✓ **Q42: Sorry, should we execute the job from within the ensemble_inputs subdirectory?**

- A: Yes, you need to launch the job script from the ensemble_inputs directory

✓ **Q43: How about tradeoff in having to queue for resources? Specifically to LUMI, would it make sense to benchmark one's system say for several GPU nodes (and how many) or stick to fewer ones to reduce queueing time at the cost of performance?**

- A: **If** your job scales well to multiple GPU nodes, then I don't see a reason for not using multiple nodes. Usually, however, you might be well off with just one node. In the end, it is nonetheless a balance you have to figure out yourself, i.e. do you want to use more time for queueing or for waiting for your job to run (using less resources).

✓ **Q44: I am not sure how I can assign tasks to multiple GPU nodes. in ex4.2 we have 16 ensembles. If we use 2 GPUs, then we would have 112 MPI ranks. Can we have 32 tasks and 2 OpenMPs?**

- A: There are only 56 cores available per GPU node, so with 32 tasks you can only use `OMP_NUM_THREADS=1`
- But we have 16 ensembles in ex4.2.
- Indeed, in that case you can use 16 tasks and `OMP_NUM_THREADS=3`
- But how can we use the most of 2 GPUs? 16 tasks and `OMP_NUM_THREADS=3` can be run in 1 GPU.
- I'm not sure I understand. Even when using 2 GPUs, you're still constrained by the total number of cores on a *node*. I.e. you still have 56 cores, not 112.
- Exactly; that is why I don't get the point of ex4.2.
- Do you mean 2 GPUs or 2 GPU nodes?
- 2 GPU nodes and 16 GCDs.
- In that case for 16 ensemble members you would run 1 task per ensemble member (i.e. per GCD) and 7 `OMP_NUM_THREADS`. That would be the best use of the resources available per node. $1 * 7 * 16 = 112$.
number of cores $\leq 1 * OMP_NUM_THREADS * \text{Number of ensembles}$.

Q45: Could you post the job script used for ex4.2?

- A: You can find all reference files and scripts here (<https://github.com/Lumi-supercomputer/gromacs-on-lumi-workshop/tree/main>)

Q46: ..?

- A:

Q47: ..?

- A:

Q48: ..?

- A:

Q49: ..?

- A:

Feedback

- Please add your feedback for the sessions/trainers
- You have also received a feedback request link by email (at around 14:10) - please give feedback and help us improve our events in the future!



Icebreaker

What are your expectations for this workshop?

- To use GROMACS on LUMI in its most optimal way.
- To get the most of the available resources in LUMI while running GROMACS
- I hope to learn how to run Gromacs on LUMI using GPUs and compare the speed to the calculations I am currently running
- I hope I can learn use of multiple GPUs (proper use of PME) in my simulation to speedup
- To make sure I'm not doing anything wrong when running

- Figure out more about LUMI and optimizing GROMACS production runs for running larger simulation boxes, larger proteins etc. (incl QM/MM).*
- To learn how to make better use of the computational resources we have recieved. Additionally, how to run more advanced MD simulations (for example T-REMD) at nice efficiencies.
- To learn running gromacs in GPU env effectively*
- To start using Gromacs on Lumi!
- Learn about LUMI and AMD GPUs in HPC
- Getting familiar with LUMI
- get an idea how I can better support our researchers w/ their computational dilemmas
- Learnin to use LUMI in the most efficient way
- See if Lumi would give me better performance with my system
- To learn how to run GROMACS on LUMI
- To see how to overcome that LUMI uses AMD
- Obtain the best performance when running MD

HackMD practice area

Try out HackMD and the Markdown syntax by typing something under here!

- Test test, this is code , **this is bold**, *this is italicized*, subscript, ^{superscript}
- $\mathbf{F} = -\nabla V$
- TEST

```
code block
with multiple
lines
```

bioexcel

The logo for 'bioexcel' features the word 'bioexcel' in a grey, lowercase, sans-serif font. The letter 'x' is replaced by a stylized human figure. The figure has a solid orange circle for a head, a green curved line for the left arm, and an orange curved line for the right arm. The two arms cross at the center to form the 'x' shape.