Programming Parallel Computers

Instructions

How to use this system?

To get started:

- Create a user account if you do not have one yet.
- Log in using the user account you just created. Please note that e.g. your university-provided user account does not work here; you will need to create a new user account just for this system.
- Select the right course. You can make it your default course so you will get automatically redirected to the right place when you come back here.

Exercises and tasks

For each course, the assignments are organized in exercises (e.g. CP) that consist of tasks (e.g. CP3b):

- Read the general description of the exercise to see what is the problem to solve. Typically, you need to write an implementation of some C++ function that solves a computational task correctly.
- Read the detailed description of the task to see exactly how you are supposed to solve the problem. In some tasks you may need to use a specific algorithm or a specific technique. For some tasks you may need to write CPU-side code in regular C++, while for some tasks you may need to write GPU-side code in CUDA.

Submitting a solution

Once you have written a correct solution to one of the tasks, here are the minimal steps for getting points for it:

- In the task page, upload the file with the solution for automatic grading.
- In a few minutes, check the results of automatic grading.
- If all looks good, you can submit it for feedback by selecting "This submission is ready for grading".
- Usually next week you will see the feedback from our course staff and how many points you got.
- If you are participating in an unsupervised course with self-study, there will be less human feedback. Submissions are only occasionally reviewed manually.

Recommended workflow

To solve a task, we recommend that you proceed as follows:

- Read both the general description of the exercise and the detailed instructions of the task. Pay attention to the time limits; in many tasks you will have to implement a very efficient solution in order to get full points.
- In the task page you can download a zip file that contains the code templates with which you can start to develop a solution. Unzip it in a suitable place.
- In the zip file you will find the grading tool that you can run with the command `./grading`. It should work fine at least in typical Linux and macOS environments.
- In the zip file you will also find a mostly-empty source file template that you are supposed to edit, using your favorite text editor. The source code template contains the function that you need to implement.
- You can try out on your own computer if your implementation works correctly, by running `./grading test`. It will try to compile your code, and it will run a selection of automatic tests that try to make sure your solution is bug-free. Simply read what the tool says, and debug the code as needed if there are bugs. Please see `./grading --help` for more information on how to use the grading tool.
- If possible, try to also benchmark your code locally by running `./grading benchmark` to get a better understanding of how fast it works (but please keep in mind that the hardware that you are using is likely very different from the hardware we use for grading).
- Once you are happy, follow the steps outlined above for the submission workflow, and upload the code for automatic grading.
- If automatic grading fails, you should get some helpful feedback on exactly which test failed, and if the input and output are small, you should also see how it failed. With this information you should be able to hopefully debug the problem locally, and try to resubmit if needed.
- If automatic grading succeeds, you should also see how fast it solved the benchmark tasks, and how many points you will get. If you are not happy with the performance, you can try to improve your code and resubmit it again.
- Once you are happy with the results of the automatic grading, and with the predicted number of points you would get, submit it for human feedback, and then move on to the next task.

Please also keep in mind that you are in no way restricted to the use of the tools that we provide. You can also write your own code that tries to run your implementation with different test inputs, you can use whatever development and debugging tools you commonly use while programming, etc. As all task in one exercise are related to the same problem, you can benefit from whatever you implement for one task also in other tasks.

If you do not have suitable hardware or software on your own computer, you can often e.g. connect to Linux computers provided by your university remotely via ssh and develop and test your code there. Another option is to send your command for remote execution, as detailed below.

Running commands remotely

Adding the `--remote` flag to any command on the grading tool, e.g. `./grading test --remote`, will send your code and the specified tests to this system. It is then run and the output is shown in your terminal once the whole command completes.

When first using the remote flag, you will be instructed to obtain an API token for your account and to copy it into an appropriate configuration file. It is easiest to place the file in the suggested system-specific configuration path so it will work with all tasks.

The feature is intended for testing and experimentation. Using the `benchmark` command remotely can be especially useful because it is run on the same hardware as normal submissions are. However, it does not replace normal submissions. Once you are satisfied with your code, remember to still make a submission through the web interface to get points.

Some of the tests are secret

The zip file with the code templates also contains some test cases, you will find the definitions of the test cases in the subdirectory `tests`. When you run `./grading test`, the script will use all of these files to ensure that your code works correctly at least with those inputs.

However, there are also some additional secret test cases that our automatic grading tool will use. So it is possible that you may have bugs that are not visible yet when you run the grading tool locally. But no worries, whenever any of our secret tests fails, you will get to see the test definition. You can even download the test definition, add it to your local `tests` subdirectory so that `./grading test` will find them, and this way you can easily reproduce the bug locally. Note that you can also specify in the command line exactly which test case to run.

Changing your mind

If you have already submitted something for grading, and then come up with a better solution before the deadline, please feel free to un-submit the previous version (choose "Please ignore this submission") and submit the better version. Please do not make multiple parallel submissions unless there is a good reason for that (e.g. you would like to understand why the improved version turned out to be slower).

Grading

For each task there is a deadline by which you need to solve it in order to get points for the task. What matters is the time when you uploaded the source code for automatic grading.

For each task, there are time limits for the running time of your solution: the number of points you get depends on how fast your implementation solves the problem for the benchmark inputs.

If you follow instructions and your solution is correct, already after automatic grading you should have a fairly good understanding of how many points you will eventually get. However, our course staff can adjust the points if needed; for example, if your implementation is almost correct but has got some minor bugs in some corner cases, you may lose one or two points.

There is no penalty for resubmissions or multiple submissions. Resubmissions are always safe; you will never lose any points you have already got. However, please also note that resubmissions are graded exactly the same way as other submissions; in particular, if you resubmit after the deadline, you cannot get full points for the resubmission.