Experimental Algorithms: Exercise sheet 1

Dr. Alexander van der Grinten

November 13, 2020

Notes

- Due by November 26, 2020, 9:00 AM.
- Each exercise is graded with 0, + or ++ (see the lecture slides).
- You may form groups of 1 - 3 participants (but groups of two are recommended).
- For exercises that require implementations, hand in the code on Moodle or by providing a public Git repository. **Make sure that your code compiles on `gruenau`!**
- **Run experiments on `gruenau3` or `gruenau4`!**
- For exercises that require you to answer questions, hand in your solution on Moodle. Use an appropriate document format, e.g., a plain text file, a PDF file, a scan, or similar.
- You can ask for clarifications regarding the exercises in the lecture or on Moodle. This also applies to technical questions regarding Simexpal, `gruenau`, understanding the code etc.
- To make sure that everything works smoothly, please ensure that you have Simexpal version 0.4. This version is released alongside this exercise sheet.
- On `gruenau`, you have to pass `--user` to `pip3` to install Python packages into your home directory. You also have to add `$HOME/.local/bin` to your `PATH`: `export PATH=$PATH:$HOME/.local/bin`

Exercise 1: Tools (Simexpal, Matplotlib)

The goal of this exercise is to get to know the tools that we will need during this course. As an example, we will use two simple hashing algorithms based on (i) chaining, and (ii) linear probing.[1] First, download the provided source code and scripts. Familiarize yourself with the C++ code in `main.cpp` (directory: `develop/hashing`) and the configuration of the experiments in `experiments.yml`.

(a) Use Simexpal to compile and run the experiments on `gruenau`. Building the experiments can be done by executing:

```
simex develop hashing
```

After the build is successful, experiments can be started by running:

```
simex e launch
```

Note that the experiments can take ≥ 15 min to finish. You can use the `simex e` command to monitor their progress (e.g., from a second terminal). If anything goes wrong, `simex e purge` can be used to

[1]The code also contains the `stl` algorithm that simply forwards to `std::unordered_map`. Interestingly, that implementation is quite slow, at least for GCC.

remove failed experiments; consult the documentation for further information. Hand in (on Moodle) an archive of the resulting output files from the `output/` directory. `simex archive` can be used to obtain such an archive.

(b) Use the provided Jupyter notebook `evaluation.ipynb` to evaluate the results. Use Matplotlib to visualize the results in two line plots: (i) one plot of insertion time vs. fill factor and (ii) one plot of lookup time vs. fill factor. More precisely, plot the fill factor on the X axis and the time on the Y axis. In each plot, add one line per algorithm. Hand in the plots as PDFs.

Exercise 2: Implementation and Profiling

In this exercise, we implement an additional algorithm and integrate it into our experimental pipeline.

(a) Implement the quadratic probing algorithm (in `main.cpp`). Note: while it is possible to test the new algorithm through Simexpal, it might be easier to simply create a separate build directory using Meson:

```
mkdir my-build && cd my-build
meson --buildtype=debugoptimized ../develop/hashing && ninja
```

(b) Run the `hashing` program manually, using the `--microbenchmark` flag and a fill factor of 0.90. This flag causes the algorithm to perform insertions only. It also switches to a less sophisticated method to generate random data; this makes sure that the majority of the running time is spent in the algorithm.

Use 'perf stat' to monitor the number of cache misses that the `chaining`, `linear` and `quadratic` algorithms cause. Interpret the results: how are the results explained by the design of the algorithm?

(c) Integrate your quadratic probing algorithm into the experiments from Exercise 1. How does it compare to the other algorithms?

Exercise 3: Robin Hood and Cuckoo

This exercise concerns the implementation of more sophisticated hashing schemes.

Details are TBA after the lecture on Monday.