Profiling NetworKit with callgrind

You can profile NetworKit using the callgrind tool of valgrind. It generates a call graph for the program annotated with call counts and optionally cache misses as well as branch-prediction misses.

Step-by-step guide

- Remove `-pg` from the build options in `SConstruct`:

  ```
  profileCppFlags = ["-O2", "-DNDEBUG", "-g"]
  profileCFlags = ["-O2", "-DNDEBUG", "-g"]
  ```

- Compile NetworKit for Profiling:

  ```
  scons --optimize=Pro --target=Tests
  ```

- Profile your tests using `callgrind`:

  ```
  valgrind --tool=callgrind ./NetworKit-Tests-Pro --tests --gtest_filter=<filter>
  ```

- Analyse the profiling results in `callgrind.out.<pid>` visually:

  ```
  kcachegrind
  ```

Compile options

- You should compile with debugging information `-g` to get a readable profile.
- We recommend compiling with optimizations `-O2` enabled, so inlining is performed and you get realistic call traces. You should also compile with `-DNDEBUG` so debug statements such as asserts are disabled.
- Do not compile with `-pg`. Otherwise you will get misleading calls to `mcount()`.

Program control

You can control the profiler in your code with the macros in the `valgrind/callgrind.h` header:

```
#include <valgrind/callgrind.h>
```

Profiling parts of the program

You can profile parts of your program by only enabling profiling in these parts:

```
CALLGRIND_START_INSTRUMENTATION;
...code to profile...
CALLGRIND_STOP_INSTRUMENTATION;
```

In this case you should also start callgrind with instrumentation disabled:

```
valgrind --tool=callgrind --instr-atstart=no ./NetworKit-Tests-Pro --tests --gtest_filter=<filter>
```

Note that this may impact the simulated cache since cache loads outside of the profiled sections are not taken into account.

Multiple profiles in a single run

You can create multiple profiles in a single run of NetworKit by dumping the current stats and resetting them to zero:

```
CALLGRIND_DUMP_STATS_AT("<name>");
```

This will create profiles `callgrind.out.<pid>.<id>` with numerical ids. Although the `name` you give will be contained in the profiles, I have not found a way to add it to the generated file names.

Cache simulation and branch prediction

You can enable cache simulation with `--cache-sim=yes` and branch-prediction simulation with `--branch-sim=yes`.

Restrictions

You should be aware that callgrind only takes the direct caller of a function into account when collecting stats, i.e. it cannot distinguish two call stacks `f1->g->h` and `f2->g->h`. Thus, some parts of the output such as the callee map may not be entirely correct.

See How profilers lie.