{{lastupdated_at}} by {{lastupdated_by}}

How to use valgrind?

Valgrid is a very useful tool for finding bottle necks in the execution of code and to identify where actually the type is spent.

Running an executable with valgrind

```
valgrind --tool=callgrind --dump-instr=yes --collect-jumps=yes gsrvy (args)
```

Running an executable on a specific code zone

Mark the code zone using some macros

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

int main()
{
  foo1();
  CALLGRIND_START_INSTRUMENTATION;
  CALLGRIND_TOGGLE_COLLECT;
  bar1();
  CALLGRIND_TOGGLE_COLLECT;
  CALLGRIND_STOP_INSTRUMENTATION;
  foo2();
  foo3();
}
```

and run valgrind as follows

```
valgrind --tool=callgrind --dump-instr=yes --collect-jumps=yes --collect-atstart=no --instr-atstart=no gsrvy (args)
```

Advanced usage

If the callgrind.out.* file is empty call

```
callgrind_control -d [hint [PID/Name]]
```

where hint is an arbitrary string you can optionally specify to later be able to distinguish profile dumps.

Check whether instrumentation is turned on using

```
callgrind_control -b
```

You can also decide to switch instrumentation on using

```
callgrind_control -i on
```

(and switching off again by specifying "off" instead of "on").

Result visualisation

Use the kcachegrind GUI to visualise the results.