Callgrind & Kcachegrind

Jiří Klepl

Profiling a proč profilovat (link na konci prezentace)

With application development, a common step is to improve runtime performance. To not waste time on optimizing functions which are rarely used, one needs to know in which parts of the program most of the time is spent.
This is done with a technique called profiling. The program is run under control of a profiling tool, which gives the time distribution of executed functions in the run. After examination of the program's profile, it should be clear if and where optimization is useful. Afterwards, one should verify any runtime changes by another profile run. [2]

Callgrind

- Profiler ve **valgrindovém** simulovaném prostředí
  - valgrind disassembluje binárky a sestavuje podle daného nástroje (zde s instrumentací)
  - základní nástroj valgrindu je memcheck, který kontroluje leaky a chybné přístupy
- Analyzuje **call-graph**; ale také i instrukce uvnitř funkcí a branch predikce, simulace cache (podle optionů)
- Odvozen z nástroje **cachegrind**, stejný formát výstupu; obojí lze zobrazit v v **kcachegrind** (nebo **qcachegrind**)

Srovnání s konvenčnějším profilingem (e.g. gprof)

Výhody

- Pohodlnost (stačí normální *release* binárka s debug flagy - není nutno compilovat se speciálními optiony (u gprof `-pg`))
- Přesnost měření a nezávislost na aktuálním zatížení stroje
- Profiling neovlivňuje aplikaci (to je u běžné instrumentace velký problém)
- Profilují se i dynamicky linkované knihovny (= profiler vidí vše)
- Lze za běhu ovládat (nemusíme trávit staletí profilingem startupu, když nás zajímá jen jedna funkce)

Nevýhody

- Pomalost (valgrind uvádí 50x slowdown)
- Limitovaná podpora méně mainstream architektur (u AMD64 není problém)

Základní použití callgrindu

```
valgrind --tool=callgrind [options] your-program [arguments]
```

your-program by měl mít debug flagy a narozdíl od debuggingu by měl být optimalizován

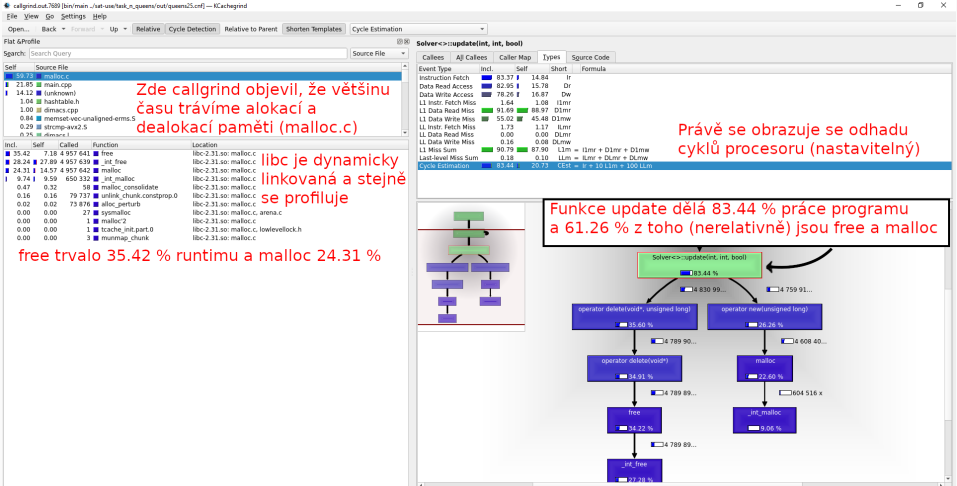
Základní optiony

- --dump-instr=yes: zanalyzuje machine code
- --collect-jumps=yes: zanalyzuje branching
- --cache-sim=yes: simulace cache

Výstup je v souboru: `callgrind.out.pid.part-threadID` (thready!)

Kcachegrind (qcachegrind)

- GUI procházení výstupu nástrojů **cachegrind** a **callgrind**
- Velice intuitivní a feature-rich s procházením kódu (a nastavitelný)



Demonstrace

Ve složce `/callgrind` uvnitř tohoto repa (link: NSWI126/callgrind)

```
cd ex1
kcachegrind callgrind.out.7689
```

```
cd ex2
kcachegrind callgrind.out.7787
```

Více informací

1. https://valgrind.org/docs/manual/cl-manual.html
2. https://www.cs.cmu.edu/afs/cs.cmu.edu/.../cl-manual.html

Díky za pozornost!