Analiza projekta korišćenjem alata za verifikaciju softvera

Seminarski rad u okviru kursa
Verifikacija softvera
Matematički fakultet

Tamara Đukić
tamarazdjukic@gmail.com

22. april 2024.

Sadržaj

1 Valgrind **2**
1.1 Memcheck 2
1.2 Callgrind 2

1 Valgrind

1.1 Memcheck

Memcheck je najpoznatiji Vagrind-ov alat i on se podrazumevano poziva ako ne stavimo dodatnu opciju koji alat pozivamo. Memcheck detektuje memorijske greške korisničkog programa kao što su:

- Curenje memorije
- Neispravno oslobađanje memorije na hipu
- Čitanje ili pisanje u nedopuštenu memoriju na hipu, steku
- Korišćenje nedefinisanih vrednosti, vrednosti koje nisu inicijalizovane ili koje su izvedene od drugih nedefinisanih vrednosti

Pokrećemo skriptu *run_ memcheck.sh* koja sadrži poziv memcheck-a. Rezultat izvršavanja se nalazi u *memcheck_ result.txt*.

```
==13690==
==13690== LEAK SUMMARY:
==13690==    definitely lost: 19,960 bytes in 29 blocks
==13690==    indirectly lost: 191,046 bytes in 294 blocks
==13690==      possibly lost: 10,465,824 bytes in 87 blocks
==13690==    still reachable: 24,602,834 bytes in 62,488 blocks
==13690==                       of which reachable via heuristic:
==13690==                         multipleinheritance: 3,440 bytes in 10 blocks
==13690==         suppressed: 0 bytes in 0 blocks
==13690==
==13690== For lists of detected and suppressed errors, rerun with: -s
==13690== ERROR SUMMARY: 11110 errors from 73 contexts (suppressed: 0 from 0)
```

Slika 1: Rezultat memcheck-a

Zaključak: Primetimo da ima dosta memorije koja je definitivno i indirektno izgubljena. Takođe imamo dosta memorije kojoj možemo da pristupimo i da je oslobodimo.

1.2 Callgrind

Callgrind je alat koji u vidu grafa generiše listu poziva funkcija korisničkog programa. Podrazumevano, prikupljeni podaci se sastoje od:

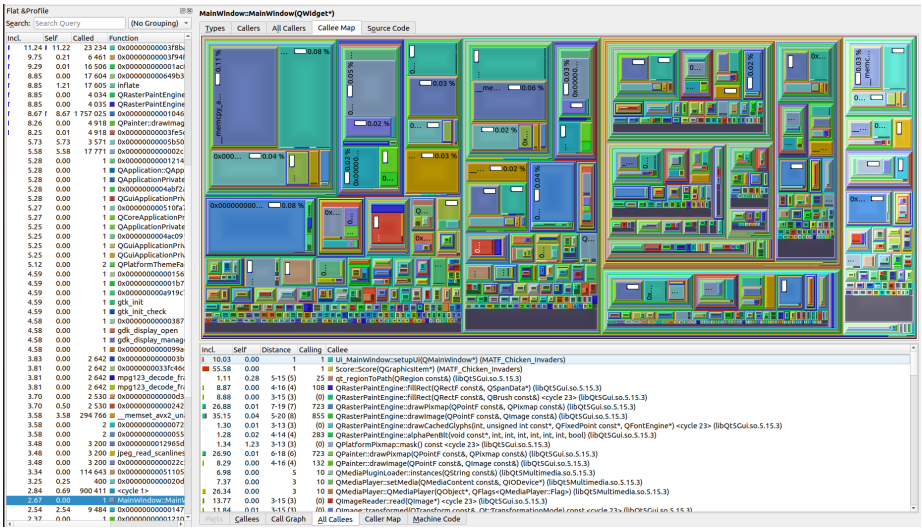
- Broja izvršenih instrukcija
- Njihovog odnosa prema izvornim linijama
- Odnos između pozivajućih i pozvanih funkcija
- Broj takvih poziva

Da bi alat Callgrind mogao da se pokrene, pre toga je projekat potrebno kompajlirati u profile režimu. Potrebno je da se kompajlira u ovom režimu da bi se izvršile dodatne optimizacije i da bi zapravo softver bio u stanju koje je slično onom za produkciju. Glavna razlika u odnosu na release režim je taj što u ovom režimu postoje dodatne debug informacije.

Pokrećemo preko skripte *run_callgrind.sh*.

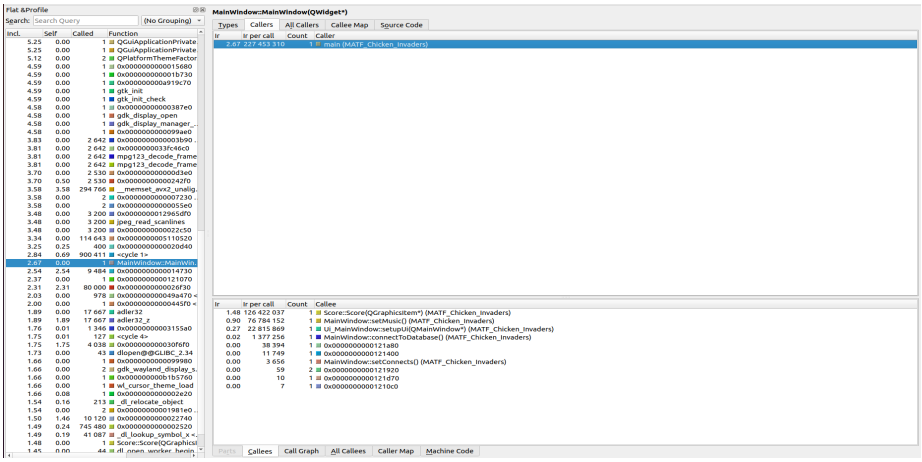
Kada se završi rad Callgrind-a, rezultati analize su zapisani u fajlu *callgrind.out.7326*. Otvorićemo taj fajl pomoću KCachegrind-a koji će nam grafički pokazati rezultate.

Prikaz Calle Map i All Calles za funkciju MainWindow(QWidget*):



Slika 2: Rezultat callgrind-a

Na levoj strani možemo videti broj pozivanja svake funkcije i broj instrukcija koje je zahtevalo njeno izvršavanje, samostalno i uključujući izvršavanja drugih funkcija koje je pozivala. Na desnoj strani smo izabrali opciju Callers koja prikazuje funkcije koje su pozivale funkciju MainWindow(QWidget*).



Slika 3: Rezultat callgrind-a

Zaključak: Posmatranjem izveštaja, može se zaključiti da nema velikog broja poziva funkcija u delu koda koji su implementirali programeri projekta.