Massif Visualizer

Seminarski rad u okviru kursa
Verifikacija softvera
Matematički fakultet

Anđela Damnjanović
mi19059@alas.matf.bg.ac.rs

24. avgust 2024.

Sadržaj

1 Opis problema **2**

2 Opis arhitekture sistema i rešenja problema **2**
2.1 Parsiranje izlaznog massif fajla 3
2.2 Vizualizacija 4

1 Opis problema

Alat Valgrind sadrži *Massif*, profajler hip memorije, te stoga meri koliko hip memorije program koji se testira koristi tokom izvršavanja. No, nisu informacije o količini memorije jedine koje Massif nudi. Ovaj alat takođe daje detaljan prikaz funkcija i mesta u kodu u kojima dolazi do alokacije memorije (takozvana *hip drveta*), što se može iskoristiti za smanjenje ukupne količine memorije koju program koristi.

Još jedna značajna prednost alata Massif je ta što pomaže u detekciji *curenja memorije* koje ni neki drugi alati specijalizovani za to ne uspevaju (kao na primer Valgrindov Memcheck).

Glavna namena programa Massif Visualizer jeste vizualizacija izlaza koji daje alat Massif. Svaki izlazni fajl sadrži informacije koje se mogu videti na slici 1. Najpre su navedene informacije o tome sa kojim je opcijama i nad kojim fajlom pokrenut alat, kao i merna jedinica vremena izvršavanja, da bi zatim usledile informacije o *snapshot-ovima* [1]. Neki snapshot-ovi su *normalni* i iza njih ne slede nikakve dodatne informacije. Sa druge strane, postoji i *detaljni* snapshot-ovi, iza kojih se nalazi opis opis izgleda stabla hipa. On obuhvata informacije o ukupnoj količini alocirane memorije, kao i podatke o tome koliko je memorije alocirala koja funkcija i u kojoj liniji koda je došlo do alokacije. Dakle, potrebno je parsirati dobijeni izlazni fajl i prikazati dobijene rezultate na grafiku.

```
desc: --max-snapshots=30
cmd: ./example
time_unit: i
#-----------
snapshot=0
#-----------
time=0
mem_heap_B=0
mem_heap_extra_B=0
mem_stacks_B=0
heap_tree=empty
```

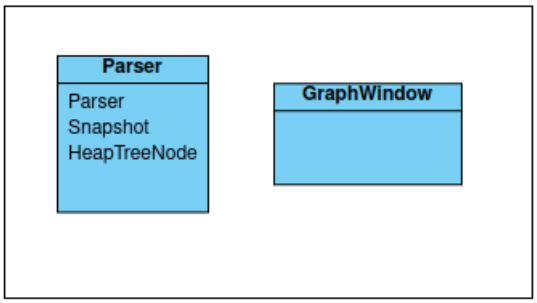
Slika 1: Sadržaj izlaznog fajla alata Massif

2 Opis arhitekture sistema i rešenja problema

Program bi se mogao podeliti na sledeće celine: parser čija je uloga da parsira prosleđene fajlove i grafički prozor koji je odgovoran za komunikaciju sa korisnikom i iscrtavanje dobijenih obrađenih rezultata. Podela se može videti na slici 2.

Sam parser je takođe podeljen na celine: parser, snapshot i hip drvo. Zadatak klase *Parser* jeste da isključivo vrši parsiranje fajla (ili fajlova) dobijenih od strane korisnika. Klasa *Snapshot* je zadužena za čuvanje informacija o svakom

[1] Snapshot predstavlja sliku memorije programa u određenom vremenskom trenutku



Slika 2: Pregled osnovnih celina alata Massif Visualizer

napravljenom snapshot-u i ujedno je i polje klase *Parser*. Klasa koja reprezentuje hip drvo zadužena je za čuvanje informacija o svakom drvetu koji se može naći u detaljnim snapshot-ovima i samim tim predstavlja i polje klase *Snapshot*.

Uloga grafičkog prozora jeste da komunicira sa korisnikom. Korisnik najpre može da bira da li želi da analizira jedan ili više fajlova ili pak želi da prevede svoj izvršni fajl sa opcijama koje sam izabere. Takođe je moguć izbor boja pozadine i grafa, kao i izbor načina na koji će dobijeni podaci biti iscrtani (običan graf ili tačkasti graf, da li će na x-osi biti podaci o vremenu ili broju snapshot-ova). Nakon odabranih parametara, poziva se parser i dobijaju se informacije koje je dalje potrebno obraditi. Nakon dobitka ovih podataka, pravi se odgovarajući graf i odgovarajući pitasti dijagrami koji se zatim prikazuju korisniku.

2.1 Parsiranje izlaznog massif fajla

Iako to nije centralna tema rada, najvažniji korak programa Massif Visualizer predstavlja parsiranje podataka iz dobijenog fajla jer ako se ovaj korak loše uradi, onda to povlači dalje greške u iscrtavanju. Stoga je najviše pažnje posvećeno upravo ovome i napravljene su pojedinačne klase za svaku od odgovarajućih komponenti, kao što je bilo naznačeno na slici 2.

Najpre je potrebno parsirati preambulu koja sadrži osnovne informacije o programu: sa kojim opcijama je pokrenut alat massif, nad kojim izvršnim fajlom i koja je vremenska jedinica korišćena pri izradi snapshot-ova (prve 3 linije na slici 1). Njihovo parsiranje nije se pokazalo izazovno. Dovoljno je bilo proveriti da li linije u fajlu redom počinju sa *desc:*, *cmd:* i *time_unit*. Ako neki od ovih uslova nije zadovoljen, prosleđeni fajl nije validan. Takođe je provereno da li linija opisa sadrži karaktere –, = i ako ne onda massif fajl nije validan, a ako je to slučaj, onda je linija dalje deljena da bi se dobili argumenti sa kojima je alat pokrenut. Slično, merna jedinica može biti označena samo sledećim skupom karaktera: *B, i, ms*. U suprotnom, fajl nije validan.

Sledi parsiranje snapshotova. Ono je urađeno na sličan način parsiranju preambule, samo sa drugačijim početnim redovima linija.

Najzad, potrebno je parsirati i dobijena hip stabla za detaljne snapshot-ove. Primer jednog takvog stabla može se videti na slici 3. Za ovu vrstu parsiranja korišćen je *regex* da bi se dobili podaci o ukupnom alociranju memorije, kao i o pojedinačnim alokacijama zajedno sa informacijama o tome u kojoj funkciji se vrši alokacija, u kojoj liniji i u kom fajlu.

```
n3: 20000 (heap allocation functions) malloc/new/new[], --alloc-fns, etc.
 n0: 10000 0x1091E5: main (valgrindExample.c:19)
 n2: 8000 0x10919A: g (valgrindExample.c:5)
  n1: 4000 0x1091B4: f (valgrindExample.c:11)
   n0: 4000 0x109201: main (valgrindExample.c:21)
  n0: 4000 0x109206: main (valgrindExample.c:22)
 n1: 2000 0x1091AF: f (valgrindExample.c:10)
  n0: 2000 0x109201: main (valgrindExample.c:21)
```

Slika 3: Izgled hip stabla

2.2 Vizualizacija

Izbori vezani za sam proces vizualizacije dobijenih podataka nisu bili značajni. Estetike radi, dozvoljeno je da korisnik bira koje boje će biti graf i pozadina, kao i debljina linija. Pošto je tema projekta vizualizacija, bilo je logično dopustiti mogućnosti iscrtavanja više vrsta grafova (običan i tačkasti graf su odabrani jer su dosta česti, a i imaju reprezentaciju u Qt-ju). Iz istog razloga dopušten je i izbor da li broj snapshot-a ili vremenska jedinica biva predstavljena na X-osi.

Dalje, za odabir opcija sa kojima se prevodi izvršni fajl delovalo je kao da je forma najbolja opcija, gde su željene opcije čekirane. Uz to je i lako za obradu, jer je jedino polje koje korisnik *mora* da popuni ono o putanji do izvršnog fajla, što olakšava obradu grešaka.

Pitasti dijagrami su takođe jako čessti za vizualizaciju podataka, te je stoga doneta odluka da se oni iskoriste. Njima je pogodno reprezentovati informacije o tome koja funkcija koliko memorije alocira, što može biti korisno ukoliko je cilj smanjiti potrošnju memorije.

Tab je kao opcija izabran jer postoji mogućnost da se njime prikaže dosta informacija, bez da se zauzme mnogo prostora.