Optimizacije kroz GCC, LLVM i Native Image

Tamara Stojković, Emilija Stošić, Teodora Isailović

Metodologija stručnog i naučnog rada
Matematički fakultet
Univerzitet u Beogradu

Beograd, decembar 2022.

Sadržaj

Uvod

- Optimizacija - tehnika transformacije dela programa, sa ciljem poboljšanja performansi koda
- Oblast u kojoj se danas vrši većina istraživanja kompajlera
- Kompajleri - različiti nivoi optimizacije
- Na nivoima kompromisi između mera - kvalitet koda, veličina koda, vreme kompilacije
- Izbor kompajlera i nivoa optimizacije zavise od konkretnog programa

Lokalne optimizacije

- Razlikujemo: lokalne, globalne i međuproceduralne
- Lokalne optimizacije služe za ubrzavanje malih delova neke funkcije, najlakše za izvođenje
- Tehnike lokalne optimizacije koje razlikujemo:
  - Eliminacija čestih podizraza
  - Slaganje konstanti
  - Propagacija kopija
  - Smanjenje snage operatora
  - Eliminacija mrtvog koda
  - Algebarsko pojednostavljenje i reasocijacija
  - Kompozicija lokalnih transformacija

Globalne i međuproceduralne optimizacije

- Globalne optimizacije primenjuju se na jednu po jednu funkciju, slične lokalnim
- Globalne optimizacije koje razlikujemo:
  - Globalna eliminacija mrtvog koda
  - Globalno propagiranje konstanti
  - Globalna eliminacija čestih podizraza
  - Optimizacija kretanje koda
  - Pomeranje invarijantog koda
  - Parcijalna eliminacija suvišnosti
- Međuproceduralna optimizacija - radi na celokupnom grafu kontrole toka
- Vrši se na nivou celog programa tj. više funkcija
- Najpoznatija tehnika je uvlačenje definicija funkcija

Optimizacije koda

- Optimizovani međukod se prevodi u asemblerski tj. mašinski kod - faza generisanja
- Bitne su specifične karakteristike mašine
- Razlikujemo sledeće tehnike:
  - Optimizacija redosleda instrukcija - obuhvata fazu odabira instrukcija, alokacije registara i raspoređivanja instrukcija
  - Optimizacija upotrebom keša - zasniva se na prostornoj i vremenskoj lokalnosti, cilj da bude što bolja

Optimizacije u okviru kompajlera GCC

- Opcije koje su vrlo značajne u procesu kompilacije, su opcije za optimizaciju
- Nivo optimizacije koju kompajler vrši se kontroliše opcijom -On, gde je n nivo zahtevane optimizacije
- Postoji osam nivoa optimizacije : -O0, -O1, -O2, -O3, -Os, -Oz, -Og, -Ofast

```
emilija@SONY:~/Desktop$ gcc -Wall -O0 primer.c -lm
emilija@SONY:~/Desktop$ time ./a.out
sum = 4e+38

real    0m1.707s
user    0m1.701s
sys     0m0.000s
emilija@SONY:~/Desktop$ gcc -Wall -O1 primer.c -lm
emilija@SONY:~/Desktop$ time ./a.out
sum = 4e+38

real    0m0.419s
user    0m0.412s
sys     0m0.000s
emilija@SONY:~/Desktop$ gcc -Wall -O2 primer.c -lm
emilija@SONY:~/Desktop$ time ./a.out
sum = 4e+38

real    0m0.381s
user    0m0.360s
sys     0m0.009s
emilija@SONY:~/Desktop$ gcc -Wall -O3 primer.c -lm
emilija@SONY:~/Desktop$ time ./a.out
```

Optimizacije u okviru kompajlera LLVM

- Optimizacije u vidu prolaza
- Svi LLVM prolazi su podklase klase Pass
- Funkcionalnosti su implementirane tako što prevazilaze virtuelne metode nasleđene od klase Pass
- Analysis Passes prikuplja informacija koje služe za otklanjanje grešaka ili vizuelizaciju programa
- Transform Passes se koriste za optimizaciju koda
- Utility Passes služe za dobijanje raznih informacija, koje su bitne za razvoj drugih prolaza

Optimizacije u okviru kompajlera LLVM

- Postoji sedam nivoa optimizacije : -O0, -O1, -O2, -O3, -O4, -Os, -Oz

```
emilija@SONY:~/Desktop$ clang -Wall -O0 main.c -lm
emilija@SONY:~/Desktop$ time ./a.out
sum = 4e+38

real    0m1.743s
user    0m1.703s
sys     0m0.008s
emilija@SONY:~/Desktop$ clang -Wall -O1 main.c -lm
emilija@SONY:~/Desktop$ time ./a.out
sum = 4e+38

real    0m0.771s
user    0m0.765s
sys     0m0.004s
emilija@SONY:~/Desktop$ clang -Wall -O2 main.c -lm
emilija@SONY:~/Desktop$ time ./a.out
sum = 4e+38

real    0m0.370s
user    0m0.369s
sys     0m0.000s
emilija@SONY:~/Desktop$ clang -Wall -O3 main.c -lm
emilija@SONY:~/Desktop$ time ./a.out
sum = 4e+38

real    0m0.364s
user    0m0.361s
sys     0m0.000s
```

Slika: Rezultati optimizacije kod LLVM kompajlera