Compiler Optimierung

Leon Ahmadi-Moghaddam
Proseminar: Effiziente Programmierung – UHH
31.05.2018

Inhalt

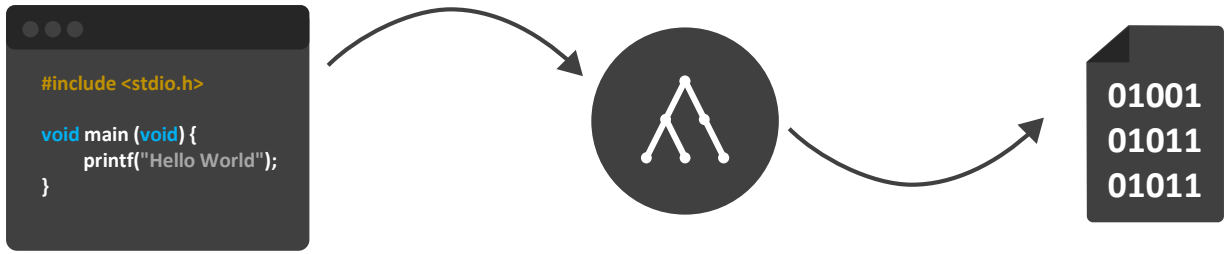
- Allgemeines zum Compiler
- Wozu Optimierungen?
- GCC
- Optimierungsverfahren Beispiele
- GCC-Optimierungslevel
- Zusammenfassung

Allgemeines zum Compiler

Was ist ein Compiler?

Allgemeines zum Compiler ● ○ ○ ○

- „System**programm**, das ein in einer **höheren** Programmiersprache formuliertes Quellprogramm (Programm) in ein Maschinenprogramm **übersetzt** (Übersetzer)“[1]



[1]https://wirtschaftslexikon.gabler.de/definition/compiler-30434/version-254016

Compileraufbau

Allgemeines zum Compiler ● ● ○ ○

Phase 1: Analyse / Frontend

Analyse des Codes und Fehlerprüfung

Phase 2: Synthese / Backend

Erzeugt den Programmcode der Zielsprache

Phase 1 - Analyse/Frontend

Allgemeines zum Compiler ● ● ● ○

- Lexikalische-, Syntaktische-, Semantische-Analyse
- Fehlerprüfung
- Syntaxbaum erstellen

Phase 2- Synthese/Backend

Allgemeines zum Compiler

- Zwischencodeerzeugung
  - Relativ maschinennaher Code
- Programmoptimierung
- Codegenerierung

Wozu Optimierungen?

Compileroptimierung

Wozu Optimierungen? ● ○ ○

- Für jeden Befehl gibt es mehrere Möglichkeiten, ihn zu übersetzen
  - Bsp.: 2 * 5 = 5 + 5
- Anforderungen an Programm
  - Speicherplatz
  - Performance

Was ist besser?

Wozu Optimierungen? ● ● ○

```
for (i=0; i < 100; i++) {
    y[i] = i;
}
```

```
y[0] = 0;
y[1] = 1;
y[2] = 2;
y[3] = 3;
y[4] = 4;
y[5] = 5;
...
y[99] = 99;
```

Space-time tradeoff

Wozu Optimierungen? ●●●

- Compiler führt Optimierungen am Code aus für…
  - … bessere Performance
  - … geringeren Speicherplatz


GCC

GNU Compiler Collection

- GCC – **G**NU **C**ompiler **C**ollection
- Compilersammlung: C, C++, Fortran, …
- Kommandozeilenbefehl für C-Compiler: **gcc**

```
gcc helloworld.c
```

Optimierungs-Flags

GCC

-fauto-inc-dec	-fforward-propagate	-fmove-loop-invariants	-ftree-ccp	-ftree-phiprop
-fbranch-count-reg	-fguess-branch-probability	-fomit-frame-pointer	-ftree-ch	-ftree-sink
-fcombine-stack-adjustments	-fif-conversion2	-freorder-blocks	-ftree-coalesce-vars	-ftree-slsr
-fcompare-elim	-fif-conversion	-fshrink-wrap	-ftree-copy-prop	-ftree-sra
-fcprop-registers	-finline-functions-called-once	-fshrink-wrap-separate	-ftree-dce	-ftree-pta
-fdce	-fipa-pure-const	-fsplit-wide-types	-ftree-dominator-opts	-ftree-ter
-fdefer-pop	-fipa-profile	-fssa-backprop	-ftree-dse	-funit-at-a-time
-fdelayed-branch	-fipa-reference	-fssa-phiopt	-ftree-forwprop	
-fdse	-fmerge-constants	-ftree-bit-ccp	-ftree-fre	

```
gcc –fdce helloworld.c
```

Optimierungsverfahren

Optimierungsverfahren

- Loop Unrolling
- Dead Code Elimination
- Common subexpression elimination
- Function inlining
- Loop-invariant code motion
- Strength-Reduction

Loop Unrolling

Optimierungsverfahren ● ● ○ ○ ○ ○ ○


```
for (int i = 0; i < 5; i++) {
    intArray[i] = i;
}
```

```
intArray[0] = 0;
intArray[1] = 1;
intArray[2] = 2;
intArray[3] = 3;
intArray[4] = 4;
```

gcc **–faggressive-loop-optimizations** program.c

Dead Code Elimination

Optimierungsverfahren ● ● ● ○ ○ ○ ○

speed 

space 

```
int return0 () {
    int b = 20;
    return 0;
    b = 24;
    printf ("a dead code-line");
    return 0;
}
```

```
int return0 () {
    return 0;
}
```

gcc **–fdce** program.c

Common Subexpression Elimination

Optimierungsverfahren ● ● ● ● ○ ○ ○

speed ↗
space ↘

```
double a = x * M_PI + c;
double b = x * M_PI + d;
```

```
tmp = x * M_PI;
a = tmp + c;
b = tmp + d;
```

gcc **–fgcse** program.c

Function Inlining

Optimierungsverfahren ● ● ● ● ● ○ ○

speed ↗

space ↗

```
for (i = 0; i < 100000; i++) {
    sum += sq(i);
}
```

```
for (i = 0; i < 100000; i++) {
    sum += i * i;
}
```

gcc **–finline-functions** program.c

Loop-Invariant Code Motion

Optimierungsverfahren ●●●●●●○


```
for (int i = 0; i > n; i++) {
    x = y + z;
    a[i] = 6 * i + x;
}
```

```
x = y + z;
for (int i = 0; i > n; i++) {
    a[i] = 6 * i + x;
}
```

gcc **–fmove-loop-invariants** program.c

Strength Reduction

Optimierungsverfahren


```
y = x * 32768;
```

```
//32768 = 2^15
```

```
y = x << 15;
```

gcc **–fivopts** program.c

GCC Optimierungslevel

Optimierungs-Flags

GCC Optimierungslevel ● ○ ○

-fauto-inc-dec	-fforward-propagate	-fmove-loop-invariants	-ftree-ccp	-ftree-phiprop
-fbranch-count-reg	-fguess-branch-probability	-fomit-frame-pointer	-ftree-ch	-ftree-sink
-fcombine-stack-adjustments	-fif-conversion2	-freorder-blocks	-ftree-coalesce-vars	-ftree-slsr
-fcompare-elim	-fif-conversion	-fshrink-wrap	-ftree-copy-prop	-ftree-sra
-fcprop-registers	-finline-functions-called-once	-fshrink-wrap-separate	-ftree-dce	-ftree-pta
-fdce	-fipa-pure-const	-fsplit-wide-types	-ftree-dominator-opts	-ftree-ter
-fdefer-pop	-fipa-profile	-fssa-backprop	-ftree-dse	-funit-at-a-time
-fdelayed-branch	-fipa-reference	-fssa-phiopt	-ftree-forwprop	
-fdse	-fmerge-constants	-ftree-bit-ccp	-ftree-fre	

```
gcc –fdce helloworld.c
```

Optimierungs-Flags

GCC Optimierungslevel ● ○ ○

-fauto-inc-dec
-fbranch-count-reg
-fcombine-stack-adjustments
-fcompare-elim
-fcprop-registers
-fdce
-fdefer-pop
-fdelayed-branch
-fdse
-fforward-propagate
-fguess-branch-probability
-fif-conversion2
-fif-conversion
-finline-functions-called-once
-fipa-pure-const
-fipa-profile
-fipa-reference
-fmerge-constants
-fmove-loop-invariants
-fomit-frame-pointer
-freorder-blocks
-fshrink-wrap
-fsplit-wide-types
-fssa-backprop
-fssa-phiopt
-ftree-bit-ccp
-ftree-ccp
-ftree-ch
-ftree-coalesce-vars
-ftree-copy-prop
-ftree-dce
-ftree-dominator-opts
-ftree-dse
-ftree-forwprop
-ftree-fre
-ftree-phiprop
-ftree-sink
-ftree-slsr
-ftree-sra
-ftree-pta
-ftree-ter
-funit-at-a-time

Level -O1

```
gcc –fdce helloworld.c
```

Optimierungslevel

GCC Optimierungslevel ● ● ○

- -O0
- -O1
- -O2
- -O3
- -Os
- -Ofast
- -Og

```
gcc –O2 helloworld.c
```

BEISPIELPROGRAMM GCCLEVELS.C

BEISPIELPROGRAMM GCCLEVELS.C

GCC Optimierungslevel ● ● ●

Statistiken

Level	-O0	-O1	-O2	-O3	-Ofast	-Og	-Os
Ausführungszeit (s) *	**8.41**	1.01	1.04	**0.45**	0.47	2.81	1.27
Dateigröße (Byte)	156 016	156 016	156 124	156 636	**157 802**	156 016	**156 124**

*(Avg. 5 Messungen, gerundet auf 2 NKS)

Zusammenfassung

Zusammenfassung

- Optimierung bezüglich Programmgröße / Ausführungsdauer
  - Space-time tradeoff (z.B. bei loop-unrolling)
- GCC Optimierungsflags (z.B. -fdce) aktivieren Optimierungen
- GCC Optimierungslevel (z.B. -O0, -O1, …) fassen mehrere Optimierungsflags zusammen
- Erhebliche Zeitersparnis durch Optimierung möglich!

Quellen

Letzter Zugriff: 27.05.2018

- https://en.wikipedia.org/wiki/Program_optimization
- https://en.wikipedia.org/wiki/Loop-invariant_code_motion
- https://en.wikipedia.org/wiki/Optimizing_compiler
- https://en.wikipedia.org/wiki/Dead_code_elimination
- https://en.wikipedia.org/wiki/Common_subexpression_elimination
- https://en.wikipedia.org/wiki/Intermediate_representation
- https://en.wikipedia.org/wiki/Space%E2%80%93time_tradeoff
- https://de.wikipedia.org/wiki/Compiler
- https://de.wikipedia.org/wiki/GNU_Compiler_Collection
- https://de.wikipedia.org/wiki/Zwischencode

Quellen

Letzter Zugriff: 27.05.2018

- https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html
- https://gcc.gnu.org/onlinedocs/gcc/Developer-Options.html
- https://www.elektronik-kompendium.de/sites/com/1705231.htm

„An Introduction to GCC – for the GNU compilers gcc and g++“
Gough, Brian J.
ISBN: 0954161793
URL: http://www.network-theory.co.uk/docs/gccintro/gccintro_45.html

Falls noch Zeit ist...

Falls noch Zeit ist...

Beispielprogramm dumpTreeExample.c

```
gcc  –O2  -fdump-tree-optimized  dumpTreeExample.c
```

- -fdump-tree-optimized