Valgrind

Praktikum „C-Programmierung“

Eugen Betke, Nathanael Hübbe,
Michael Kuhn, Jakob Lüttgau, Jannek Squar

Wissenschaftliches Rechnen
Fachbereich Informatik
Universität Hamburg

2018-12-10

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Was ist ... ? [NS07b]

- *"Valgrind is a dynamic binary instrumentation (DBI) framework"*
- Valgrind core + tool plug-in = Valgrind tool

  Core: Instrumentierung der Client-Applikation

  Plug-In: Durchführung der Analyse
- Einfache Bedienung
  - Sourcecode unverändert
  - Robust und weit verbreitet
  - Vielseitige Kontrollmöglichkeiten
- Kombinierte Ausführung mit gdb möglich
- Open Source (GNU General Public License, version 2)

Funktionsweise [NS07b]

- Dynamic Binary Instrumentation $\rightarrow$ Arbeit auf Maschinencode
  - Kein Neukompilieren notwendig
  - Kein Zugriff auf Source-Code notwendig
- Längere Laufzeit durch Emulation
  - Simuliert *Gast*-CPU
  - Shadow Values ("Software-Wrapper") [NS07a]
  - Kernel bleibt Black-Box $\rightarrow$ Systemcalls teuer
  - Serialisiert Threads

  $\Rightarrow$ meist schlechtere Laufzeit um mindestens eine Größenordnung

Ablauf [NS07b]

1. Laden: Valgrind-Core, Tool, Client-Applikation
   - Valgrind-Subsysteme: Address-Space-Manager, interner Memory-Allocator, ...
   - Client-Informationen über eigenen Program-Loader einlesen
2. Anwendung von *Disassemble-and-Resynthesise* auf Code Blöcke[1]
   a) Disassemble Block-Maschinencode
   b) Instrumentiere IR [2]
   c) Rekompiliere angepassten IR
3. Ausführung zusammenhängender Code-Blöcke

[1] ca. 50 Instruktionen oder Branches
[2] Tool-Plugin

Valgrind-Tools

- **Memcheck** (Test auf Speicherfehler)
- Cachegrind (Statistik über Cache-Nutzung)
- **Callgrind** (Erstellt Call-Graph)
- **Massif** (Statistik über Heap-Nutzung)
- Helgrind (Test auf Race Conditions)
- DRD (Test auf Fehler beim Multithreading)

Verwendung

Kompilieren:

```
1 $ gcc -g -o app app.c
```

Ausführen:

```
1 $ valgrind --tool=<tool> [valgrind-options] ./app [app-options]
```

Hinweis

- Kompilieren mit `-g` Flag nicht vergessen!
- Hilfe: `man valgrind` oder `valgrind --help`

Motivation aus *Memory (Stack and Heap)*

Fehlerhafte Speicherverwendung:

- Memory leak
- Verwendung nach `free`
- `free` auf nicht-dynamischen Speicher
- Zugriff auf nicht-allokierten Speicher
- Mehrfache Speicherfreigabe mit `free`

Außerdem:

- Verwendung nicht-initialisierter Werte
- ...

Motivation [Manc]

- Default-Tool von Valgrind
- Manuelle Verwaltung von dynamischen Speicher in C
  - Vorteil: User kann Speicher-Zugriffe optimieren
  - Nachteil: User kann sich das Leben beliebig schwer machen

⇒ Memcheck prüft Programm auf fehlerhafte Speicher-Nutzung

```
1 $ valgrind ./app [app-options]
```

DEMO

Motivation [Mana]

- Visualisierung des Programmablaufs
- Erkennung potentieller Bottlenecks
- Genauer als einfaches Sampling
- Relativ großer Overhead zur Laufzeit kurzer Methoden
- Ergebnisse in `callgrind.out.<pid>`
- Programme zur Auswertung
  - Texteditor (nicht!)
  - `callgrind_annotate`
  - `kcachegrind`

```
1 $ valgrind --tool=callgrind ./app [app-options]
```

DEMO

Motivation [Manb]

- Misst Speicherverbrauch
  - "Wirklich verwendeter"Heap-Speicher
  - Indirekt verwendeter Heap-Speicher (Metadaten, alignment, usw.)
  - Stack (nicht per default)
- Ergebnisse in `massif.out.<pid>`
- Rückgriff auf gdb notwendig bei oom
- Programme zur Auswertung:
  - `ms_print`
  - `massif-visualizer`

```
1 $ valgrind --tool=massif [--time-unit=B] ./app [app-options]
```

DEMO

Was haben wir gelernt?

- Speicherfehler schnell gemacht
- Speicherfehler sind böse
- Valgrind ist ein vielseitiges Tool ...
- ... beeinträchtigt aber die Laufzeit negativ
- Einfache Anwendung

Quellen I

[Mana] Valgrind User Manual. Callgrind: a call-graph generating cache and branch prediction profiler.

[Manb] Valgrind User Manual. Massif: a heap profiler.

[Manc] Valgrind User Manual. Memcheck: a memory error detector.

[NS07a] Nicholas Nethercote and Julian Seward. How to shadow every byte of memory used by a program. In *Proceedings of the 3rd international conference on Virtual execution environments - VEE '07*. ACM Press, 2007.

[NS07b] Nicholas Nethercote and Julian Seward. Valgrind. In *Proceedings of the 2007 ACM SIGPLAN conference on Programming language design and implementation - PLDI '07*. ACM Press, 2007.