Debian Code Search

Michael „sECuRE“ Stapelberg

2013-06-01, GPN13

Inhalt

- Motivation
- Wie funktioniert eine Suchmaschine?
- Grundlage: Russ Cox' Codesearch tools
- Debian-Kontext
- Architektur
- Invertierter Index
- Ranking
- Optimierung
- Fragen?

Motivation: Warum Code suchen?

- Code-Beispiele statt schlechter Dokumentation
  (z.B. OpenSSL, XCB, ...)
- Einfach Projekt-übergreifend Programmfluss verfolgen
- ...

Warum eine neue Suchmaschine?

- sprach-spezifisch (Nullege, Sourcerer)
- nicht aktuell/enthält wenig Software (Koders, ohloh code, Krugle)
- nicht (mehr) verfügbar (Google Code Search)
- kein FOSS-Projekt
- abgesehen davon: interessant!

Wie funktioniert eine Suchmaschine?

- **Corpus**: Datenmenge, die durchsucht wird
- **Query**: Anfrage an die Suchmaschine
- **Inverted index**: Datenstruktur für Query $\rightarrow$ Corpus
- **Ranking**: Bestimmt Reihenfolge der Ergebnisse

Grundlage: Russ Cox' Codesearch tools

- Russ Cox hat 2006 Google Code Search implementiert
- Artikel und Beispiel-Implementation von Regular Expression Matching auf große Datenmengen
- *kurze Demo*

Debian-Kontext

- Freie, nicht-kommerzielle Linux-Distribution
- Debian Developer seit März 2012
- Viel Software: $\approx 17\,000$ Pakete, 129 GiB Code
- Nur freie Software, kein Crawler nötig
- Maschinenlesbare Metadaten, Nutzungszahlen

Architektur: High level

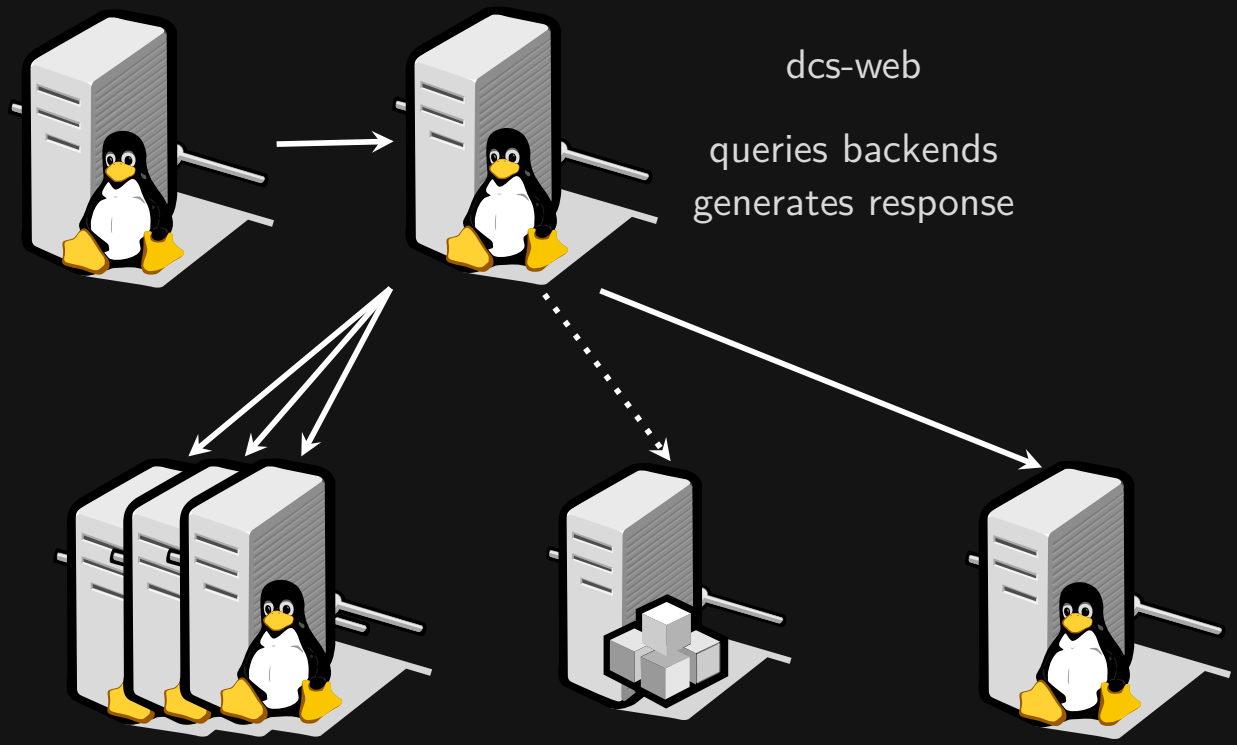
Erklärung anhand kurzer Demo

Einschub: Häufigste Queries

search term	hits	cached	cache ratio
The Software shall be used for Good, not Evil	2066	1657	80.2 %
fuck	1247	423	33.9 %
workaround package:linux	683	128	18.7 %
XCreateWindow	528	71	13.4 %
idiot	265	8	3.0 %
AnyEvent::I3 filetype:perl	255	35	13.7 %
shit	130	10	7.7 %
FIXME	116	30	25.9 %
B16B00B5	105	45	42.9 %
(babefee1|B16B00B5|0B00B135|deadbeef)	94	38	40.4 %

Architektur: Komponenten

HTTP frontend
(nginx)
delivers static assets
load-balances requests



dcs-web

queries backends
generates response

index backends (sharded)

PostgreSQL

source-backend

Invertierter Index

- `d1.txt`: Das Wetter ist schön.

- `d2.txt`: Die Wetter-Vorhersage lügt.

- Index (Begriff $\rightarrow$ posting list):
  Das: d1.txt
  Wetter: d1.txt, d2.txt
  ist: d1.txt
  schön: d1.txt
  Vorhersage: d2.txt
  lügt: d2.txt

- Ganze Wörter, keine Satzzeichen

Invertierter Index (2)

- „klassischer“ invertierter Index nicht nutzbar für Regular Expression-Suche
- `.*bar`: nach „bar“ suchen liefert nicht „foobar“
- `..._free`: nach allen Kombinationen suchen? (Unicode!)
  aaa_free, aab_free, ...
- Stattdessen: Anzahl zu durchsuchender Dateien minimieren

Invertierter Index (3)

- Dateien in 3-Gramme zerlegen
  `XCr: x.c`
  `Cre: x.c, font.c`
  ...
  (`XCreateWindow` in x.c, `CreateFont` in font.c)

- Regular Expression in Trigramm-Anfrage wandeln
  `/Debian.*Code/` $\rightarrow$ `Deb` *AND* `ebi` *AND* `bia` *AND* `ian` *AND* `Cod` *AND* `ode`

- 2-Gramme: zuviele gemeinsame 2-Gramme (Index aussagelos)
  4-Gramme: zuviele eindeutige 4-Gramme (Index zu groß)

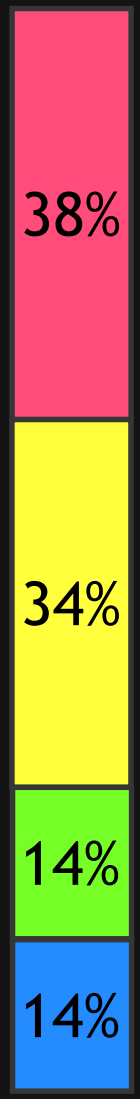
Demo: Query zu Ergebnisliste

Demo: Query zu Ergebnisliste

Ranking

- Welche Dateien sollen zuerst durchsucht werden?
- Welche Ergebnisse sollen zuerst angezeigt werden?
- Evaluation über Suchanfragen und erwartete Ergebnisse
  Beispiel: „`XCreateWindow`“ und
  `libx11_1.5.0-1/include/X11/Xlib.h:1644`
  `libx11_1.5.0-1/src/Window.c:100`

Ranking: Gewichtete Summe



Popularity Contest

Reverse Dependencies

Dateipfad

Rest:
Scope, Exact Word Match,
Package Name Match

Optimierung: Posting list decoding

- Posting lists nutzen variable integer delta encoding:
  450, 452 $\rightarrow$ `0x83 0x42`, `0x02`

- Posting list decoding re-implementiert in C

Optimierung: Query Planner

trigram	$\#P_i$	i	$\#\cap$	$\Delta_{P_{i-1}}$	Δ_R	t_{decoding}
Xcr	763	1	763		503	12 µs
teW	5732	2	266	497	6	38 µs
eWi	15 568	3	263	3	3	58 µs
Win	46 968	4	261	2	1	283 µs
Cre	78 955	5	260	1	0	209 µs
dow	97 453	6	260	0	0	344 µs
ndo	107 002	7	260	0	0	249 µs
eat	192 107	8	260	0	0	573 µs
ind	234 540	9	260	0	0	513 µs
rea	299 415	10	260	0	0	813 µs
ate	419 943	11	260	0	0	896 µs

Fazit

- Im Alltag bewährt, für andere Entwickler nützlich
- Patches welcome! Ideen vorhanden
- Zufällig 8 GiB RAM und 160 GiB SSD zuviel?

Fragen?

URLs

- http://codesearch.debian.net/
- http://codesearch.debian.net/research/bsc-thesis.pdf
- http://swtch.com/~rsc/regexp/regexp4.html