Tracen mit dem Linux-Kernel

Jürgen Stuber

2017-04-17

Ziele

- System im laufenden Betrieb beobachten
- Debuggen von Performance-Problemen
- Debuggen allgemein

ohne zu viel Overhead an die relevanten Informationen kommen

Verwirrung

- über $>$ 15 Jahre historisch entwickelt
- ein Etikett auf mehreren Dingen aus einer Quelle
  ftrace
  - function tracer
  - Infrastruktur (Ringpuffer, debugfs-Anbindung)
- mehrere Namen für das Gleiche im Laufe der Entwicklung
  perf_events
  perf
  Performance Counters for Linux (PCL)
  Linux Perf Events (LPE)
- Noch ziemlich neu, größere Erweiterungen in Linux 4.x

Orientierung an der Funktion

- Einsprungpunkte für das Tracen (Probes)
- mögliche Aktionen (Events)
- Ausgabe und Analyse von Daten

Einsprungpunkte im Code

Ziele

- geringer Overhead
- flexibel

Hauptvarianten

- ftrace
- tracepoints, *probes

ftrace

ftrace: ein Tracer für viele Funktionen gleichzeitig

alle Funktionen möglich (außer durch inlining verschwundene)

Tracer sind vorgegeben:

nop: abgeschaltet
function: Loggt Einsprung in Funktion, mit Aufrufer
function_graph: Loggt Einsprung und Rücksprung

gut für die Beobachtung des Programmablaufs

Implementierung:
GCC-Profiler-Aufrufe, wird bei Boot durch NOPs gepatcht

deaktiviert $\rightsquigarrow$ Overhead vernachlässigbar

vor Aktivieren erst Filter für Funktionen setzen

Demo

tracepoints, *probes

- Event erzeugen (Event-Infrastruktur, s.u.)
- eigenen Code aufrufen
  - registrieren durch Kernel-Modul
  - Live-Patching möglich

tracepoints

ca 500 feste Tracepoints im Kernel-Code

erzeugen Events

NOPs wenn deaktiviert ("static keys")

werden über Macros im Kernel-Code definiert:
TRACE_EVENT: definiert Tracepoint & Event (Normalfall im Kernel) DECLARE_TRACE: nur Tracepoint

eigentlicher Tracepoint ruft Funktion auf:

```
static inline void trace_##name()
```

kprobes

kprobe auf jeder Maschineninstruktion im Kernel möglich

ersetzt Instruktion durch Breakpoint
single-step durch Originalinstruktion anderswo ($\sim 1\,\mu$s)
Interrupts abgeschaltet

Optimierung durch Jump wo möglich ($\sim 100\,$ns)

außer Tracen auch Live-Patching möglich

in Rohform schwer zu bedienen $\rightsquigarrow$ Tools wie SystemTap

andere probes

uprobe: Wie kprobe aber im User-Space

kretprobe, uretprobe: Rücksprung zu gegebener Funktion
Implementierung:
ersetzt Rücksprungadresse durch Trampolin-Adresse trampolin macht Probe
Sprung zu eigentlicher Rücksprungadresse

jprobe: einfacher Zugriff auf Funktionsargumente

debugfs

mount -t debugfs debugfs /sys/kernel/debug Tracing in /sys/kernel/debug/tracing Ausgabe /sys/kernel/debug/tracing/trace /sys/kernel/debug/tracing/trace_pipe

Events

- debugfs: `/sys/kernel/debug/tracing/events`
- Events im Kernel:
  - ca 500 tracepoints vorgegeben
  - ca 500 System-Call Enter/Exit
  - kprobe-Events mit debugfs definierbar
  - einzeln aktivier- und konfigurierbar
  - debugfs: `events/<subsystem>/<event>/`

Events

- diverse aktionen konfigurierbar in
  `events/<subsystem>/<event>/trigger`
  - default: Daten in Ringpuffer speichern
  - traceon/traceoff: Trace ein/aus
  - enable_event/disable_event: anderen Event ein/aus
  - stacktrace
  - snapshot: bei Problemsituation Schnappschuss, dann Trace analysieren
  - hist (ab 4.7): Histogramme von Daten bei Events
  - enable_hist/disable_hist: Histogramm für anderen Event ein/aus
- Filter
  Beispiel: Filtern nach PID
- Anzahl der Ausführungen begrenzen
- `Documentation/trace/events.txt`

Sonstiges

- Kommandozeilentool: `trace-cmd`
  dünner Wrapper um `/sys/kernel/debug/tracing`
- perf: Tool, um Performance Counter der CPU zu lesen
  Profiling:
  Welcher Code verbraucht die Rechenzeit?
  Effekte von Caching/Branch-Prediction usw. analysieren

  Beispiel: `$ perf stat <command>`
- eBPF: Berkeley Packet Filter BPF) $\rightsquigarrow$ VM im Kernel
  Datenanalyse schon im Kernel
  spart Kontext-Switches
  sicherer als C-Code
  ab Linux 4.9 vollständig (Funktionsumfang wie Sun DTrace)

Docs

- `Documentation/trace`
- Brendan Gregg (Beispiele, Marketing)
  `http://www.brendangregg.com/blog/index.html`

  Ponies by Deirdré Straughan and General Zoi's pony creator

Ende