Fakultät für Informatik und Mathematik

HOCHSCHULE
FÜR ANGEWANDTE
WISSENSCHAFTEN
MÜNCHEN

Betriebssysteme, SoSe 2020

Praktische Übung 2: Userspace Tracing

Ziele des Labortermins

- Tracing verstehen und vergleichen

Aufgabe 2.1: LTTNG Userspace Tracing

In den kommenden Übungen wollen wir Tracing mit LTTNG (`https://lttng.org/`) verwenden um die Ausführung Ihrer Prozesse im Kernel und im Userspace zu beobachten. Tracing erlaubt Ihnen dabei die Ausführung Ihres Programms zu beobachten, *ohne* dabei auf seine Ausführung Einfluß zu nehmen. Dies ist insbesondere dann sehr wichtig, wenn Sie Probleme der Parallelität debuggen wollen. Sobald Sie auf die Ausführung durch eine interaktive Debug-Session Einfluß nehmen sehen Sie das Problem unter Umständen nicht mehr. Daneben wird Tracing intensiv eingesetzt um Systeme zu beobachten, um zum Beispiel ihre Performance zu evaluieren.

Tracing bedeutet, dass während der Ausführung einer Software (Tracing gibt es aber auch für Hardware) Ereignisse aufgezeichnet werden. Diese "Trace Events" müssen dabei von der Software ausgegeben und dem Tracing-Laufzeitsystem aufgezeichnet werden. Damit Ihre Software selbst Traces ausgeben kann, müssen Sie diese "instrumentieren": Sie fügen bestimmte Punkte ein an denen das Ereignis ausgegeben werden soll (Beispiele: Beginn und Ende einer Berechnung). Die einfachste Art des Tracing ist das beliebte "printf-Debugging" bei dem Sie Informationen während der Laufzeit ausgeben. Der Aufruf der printf-Methode, die damit verbundenen Berechnungen und insbesondere ein System Call verzögern dabei dabei die Ausführung Ihrer Software. Ein Tracing-System wie LTTNG versucht hingegen Ereignisse "minimal-invasiv" zu machen, so dass diese keinen signifikanten Einfluss auf die Ausführung Ihrer Software haben.

Dies hat zum Beispiel auch den Vorteil, dass Sie die Tracing Ereignisse (oder Tracepoints) in der Software belassen können und dann jeweils entscheiden ob Sie sie aufzeichnen und auswerten wollen. Im Bereich der Zertifizierung von Software spielt dieser Aspekt zum Beispiel eine Rolle.

Machen Sie sich mit dem Konzept des Tracing am Beispiel von LTTNG vertraut, indem Sie insbesondere folgende Themen erarbeiten:

a) Grundlegende Konzepte (Session, Domain, Channel, Instrumentierung): `https://lttng.org/docs/v2.10/#doc-core-concepts`

b) Komponenten von LTTNG: `https://lttng.org/docs/v2.10/#doc-plumbing`

c) Instrumentierung Ihrer Software: `https://lttng.org/docs/v2.10/#doc-instrumenting`

d) Ausführung einer Tracing Session: `https://lttng.org/docs/v2.10/#doc-controlling-tracing`

Aufgabe 2.2: LTTNG Userspace Tracing anwenden (praktisch)

Anhand eines einfachen Beispiels in `measure` wollen wir uns im Folgenden die Vorteile von Userspace-Tracing praktisch anschauen. Ziel ist es LTTNG Userspace Tracing (UST) Tracepoints zu verwenden und ihren Einfluss auf die Ausführungsdauer zu messen. Dies hat sehr praktischen Nutzen für die Bearbeitung der zukünftigen Übungen.

a) Vervollständigen Sie die Datei `main.c` dahingehend, dass die Dauer der `printf`-Funktion gemessen und ausgegeben wird. Dafür stehen in in `measure.h` bereits die passenden Funktionen zur Verfügung (`measure_start()`, `measure_end()` und `measure_print()`).

b) In der Header-Datei `tp.h` ist bereits ein Tracepoint definiert, der den Zählerwert ebenso ausgeben soll wie das printf. Verwenden Sie diesen Tracepoint entsprechend der Anleitung von LTTNG-UST in der Funktion um den Zählerwert zusätzlich "zu tracen". Messen Sie hier ebenfalls die Ausführungsdauer und vergleichen Sie beide Werte.

c) Führen Sie nun das Programm noch einmal aus innerhalb einer LTTNG Tracing Session. Wie verändert sich die Ausführungsdauer?

d) Werten Sie nun den Trace aus. Dafür verwenden Sie das Programm `babeltrace <pfad/zu/trace>`, dass den Trace textuell ausgibt.

e) Öffnen Sie die GUI "Trace Compass" und öfffnen Sie Ihren Trace darin. Sie finden dort ebenfalls den Tracepoint in einer Liste als auch auf einem Zeitstrahl (in diesem Fall nicht sonderlich interessant).