# LockDoc: Trace-basierte Lock-Analyse im Linux-Kernel

*Alexander Lochmann*, Horst Schirmeier, Hendrik Borghorst und Olaf Spinczyk

30. Juli 2018

Der Linux-Kernel wurde ursprünglich nicht für feingranulares Locking entwickelt. In der Frühzeit des Linux-Kernels existierte lediglich ein globales Lock, das sogenannte *Big Kernel Lock*. Dieses Lock sicherte den Zustand des Kerns gegen konkurrierende Zugriffe ab. In den letzten Jahrzehnten ist jedoch ein Trend hin zu Many- und Multicore-System zu verzeichnen. Da die Performanz mit einem globalen Lock auf Mehrkernsystemen nicht entsprechend skaliert, entstand eine Vielzahl an Locking-Primitiven, um ein feingranulareres Locking zu realisieren.

Bücher wie „Linux Kernel Development 3rd Edition“ von Robert Love geben einen umfassenden Einblick in die verschiedenen Primitiven. Dies beinhaltet sowohl die Programmierschnittstelle als auch eine Beschreibung der Kontexte, in denen sie eingesetzt werden können. Allerdings enthält die einschlägige Literatur über Locking-Primitiven kaum bis keine Informationen darüber, welches konkrete Lock für den Zugriff auf eine Datenstruktur des Kerns zu verwenden ist.

Der Linux-Kernel bietet hier ebenfalls keine zentrale Stelle, wo solche Locking-Regeln niedergeschrieben werden. Stattdessen ist diese Information über den gesamten Quellcode verteilt. Darüber hinaus ist nicht zwingend jedes Element einer Datenstruktur dokumentiert. Im besten Fall finden sich die dokumentierten Regeln als Kommentar in der Nähe der Deklaration der jeweiligen Datenstruktur, wie z.B. in dem Unterverzeichnis include/. Andernfalls muss ein Kernel-Entwickler in den zentralen Quellcodedateien für die jeweilige Datenstruktur nachschauen. Für die Datenstruktur struct inode finden sich diese Regeln z.B. in den ersten Zeilen der C-Dateien im Verzeichnis fs/.

Selbst wenn ein Linux-Entwickler eine entsprechende Regel gefunden hat, stellt sich oft heraus, dass sowohl die Beschreibung der Regel selbst als auch die Benennung des Locks informellen Charakter besitzt. Zur Beschreibung einer Regel werden oftmals unterschiedliche Wörter verwendet. Zusätzlich wird anstatt des Namens der Locking-Variable eine Prosa-Beschreibung selbiger genutzt, die möglicherweise zum Zeitpunkt der Dokumentation eindeutig genug war. Außerdem wird die Dokumentation der Regeln während der Weiterentwicklung des Kerns nicht selten vergessen, so dass diese schlicht nicht mehr aktuell sind.

Anhand der existierenden Dokumentation in Form von Quellcode-Kommentaren ist abzulesen, dass selbst erfahrene Linux-Entwickler nicht sicher sind, welche

Locks an einer bestimmten Stelle wirklich erforderlich sind. Daher wird an manchen Stellen (zu) pessimistisch gesperrt; an anderen Stellen wiederum ist es möglich, dass zu wenige Locks verwendet werden.

Zusammenfassung ist zu sagen, dass die Locking-Regeln nur teilweise und nicht gut dokumentiert sind und die Dokumentation manchmal falsch oder inkonsistent zum eigentlichen Quellcode ist. Letzteres kann bzw. hat sehr wahrscheinlich schon zu Fehlern geführt, die im Zusammenhang mit Locking stehen.

Daher präsentieren wir in dieser Arbeit einen Ansatz, um die obengenannten Probleme zu beseitigen: Mit Hilfe einer virtuellen Maschine zeichnen wir Speicherzugriffe und Lock-Operationen in einem instrumentierten Linux-Kernel auf. Anhand der Aufzeichnung rekonstruieren wir den Programmablauf und analysieren die beobachteten Lockkombinationen für Lese- und Schreibzugriffe auf jedes Element einer Datenstruktur. Die so gewonnen Ergebnisse nutzen wir, um zu überprüfen in welchem Maß die existierende Dokumentation zum Quellcode passt, um neue Dokumentation zu generieren und um potentielle Fehler zu finden. Wir zeigen also Speicherzugriffe auf, die nicht einer bestimmten Locking-Regel folgen.