Semi-Extended Tasks: Geteilter Stack für selbstblockierende Fäden

Christian Dietrich, Daniel Lohmann

July 20, 2018

Hauptspeicher ist in tief-eingebetteten Systemen eine teure und, daher, sehr begrenzte Ressource. Überschreitet eine Anwendung den angebotenen Speicher eines Mikrocontrollers auch nur geringfügig, so ist es meist nötig den nächst größeren Mikrocontroller aus der Serie mit doppeltem RAM zu verwenden. Da solche tief-eingebetteten Systeme häufig in großen Stückzahlen produziert werden, summieren sich selbst geringfügige Mehrkosten zu enormen Summen. So macht ein einzelner Cent pro Fahrzeug einen Unterschied von 110.000 Euro für Volkswagen im Jahr 2017.

Ein bedeutender Faktor für den statischen Speicherverbrauch von ereignisgesteuerten Echtzeitsystemen sind die Ausführungsstacks der einzelnen Fäden. Dabei muss jeder Stack für den maximalen Stackverbrauch des assoziierten Fadens dimensioniert und dauerhaft alloziert werden, selbst wenn dieser maximale Stackverbrauch niemals in allen Fäden gleichzeitig auftritt. Um das Problem der statischen Stackallokation zu mindern, wurde im OSEK OS Standard das Konzept der Basic Tasks eingeführt, welche alle auf einem gemeinsamen, geteilten Stack ausgeführt werden können. Dies wird dadurch ermöglicht, dass Basic Tasks nur durch höher-priore Fäden verdrängt werden, sich selbst aber niemals selbst bis zur Ankunft eines Ereignisses schlafen legen können. Durch diese Beschränkung kann es niemals dazu kommen, dass ein Basic Task, der "unten" im Stack liegt, aufwacht und den Stackspeicher eines anderen Fadens überschreibt. Basic Tasks liegen strikt nach ihren Prioritäten sortiert auf dem gemeinsamen Stack.

Mit dem Konzept von Semi-Extended Tasks lockern wir die Einschränkung der Selbstblockade, sodass auch blockierende Fäden den gemeinsamen Stack nutzen können, und dadurch der Speicherverbrauch des Gesamtsystems geringer ausfällt. Dies erreichen wir dadurch, dass wir den Aufrufgraphen eines Fadens statisch analysieren und Untergraphen identifizieren, die niemals schlafen können. Mittels eines modifizierten Übersetzers und eines leicht angepassten Betriebssystems, transferiert sich der Faden für die Ausführung des nicht-schlafenden Untergraphens auf den gemeinsamen Stack. Nur für Funktionen, die potenziell blockieren können, muss noch ein privater Stack, der nun kleiner dimensioniert werden kann, für den Semi-Extended Tasks vorgehalten werden.

Für unsere synthetischen Benchmarks könnten wir den Stackverbrauch in 80 Prozent der Fälle, gegenüber der reinen Basic Task Methode, verbessern und im Durchschnitt um weitere 7 Prozent verringern.