6 Übungsaufgabe #6: Fadenumschaltung und Synchronisation

Im Rahmen von Aufgabe 6 wird der einfache Fadenwechsel aus Aufgabe 5 erweitert. Über eine Konfigurationsoption in *KConfig* soll eingestellt werden können ob eine gemeinsame Bereitliste für alle Kerne verwendet wird, oder, wie bisher, `N` Bereitlisten für `N` virtuelle Prozessoren vorhanden sind.

6.1 Konfigurierbares Auswechseln der Schedulerimplementierung

Faktorisiert eure Implementierung aus Aufgabe 5 in Aspekte, so dass diese mit Hilfe von *KConfig* als Merkmal konfiguriert werden kann. Zusätzlich soll noch eine Variante des Schedulers mit nur einer einzigen Bereitliste implementiert werden. Zwischen beiden Varianten soll in *KConfig* als Alternative ausgewählt werden können.

6.2 Verdrängungssperre und Spinlocks

Implementiert eine Verdrängungssperre analog zu der in der Vorlesung vorgestellten Variante. Überlegt euch für welche Datenstrukturen diese Art der Synchronisation ausreichend ist, und wendet sie dort an.
Kritische Abschnitte, die nicht mit einer Verdrängungssperre gesichert werden können, sollen mit Hilfe einer Spinlock-Implementierung abgesichert werden.

6.3 Spinlock-Implementierung

Es sollen zwei Varianten eines Spinlocks implementiert werden:

- Eine, bei der die meiste Zeit auf einer lokalen Variable gewartet wird, um Cachekohärenznachrichten möglichst einzudämmen
- Eine, die zwischen zwei Umläufen eine einstellbare Zeit wartet, bevor sie wieder versuchen wird die Sperre zu nehmen.

Der dann zur Synchronisation verwendete Spinlock soll über *KConfig* auswählbar sein.

6.4 Ticket-Spinlocks (optional für 7.5 ECTS)

Zusätzlich zu den in in 6.3 genannten Spinlockvarianten soll noch ein Ticket-Spinlock implementiert und in die Konfiguration miteinbezogen werden, so dass man dann letztlich zwischen drei verschiedenen Spinlock-Varianten wählen kann.

6.5 Unterbrechungstolerante Fadenumschaltung (optional für 7.5 ECTS)

In der Vorlesung wurde eine Möglichkeit vorgestellt, den Umschaltvorgang zwischen zwei Fäden tolerant gegenüber Unterbrechungen zu machen. Dies basiert darauf den Prozesszeiger auf den Stapelzeiger abzubilden, so dass beim Fadenwechseln nur noch ein Zeiger umgeschaltet werden muss. Implementiert dieses Verfahren, um die Fadenumschaltung nicht mehr sperrend schützen zu müssen. Dabei ist es notwendig ausgerichtete Stacks zu verwenden. Dies realisiert man am besten, indem man mit `mmap(2)` einmal einen größeren Block Speicher vom Host-Betriebssystem holt und dann aus diesem händisch die Stacks für die einzelnen (Anwendungs-)Fäden entsprechend ausgerichtet allokiert.

Aufgaben:

→ Schedulervarianten mit globaler und verteilter Bereitliste als Aspekterweiterung implementieren und mit *KConfig* konfigurierbar machen.

→ Selektive Verdrängungssperre implementieren und wo möglich zur Synchronisation verwenden.

→ Implementieren der verschiedenen Spinlock-Derivate

→ Übrige kritische Abschnitte mit einem konfigurierbaren Spinlock sichern

→ Dispatcher unterbrechungstolerant machen