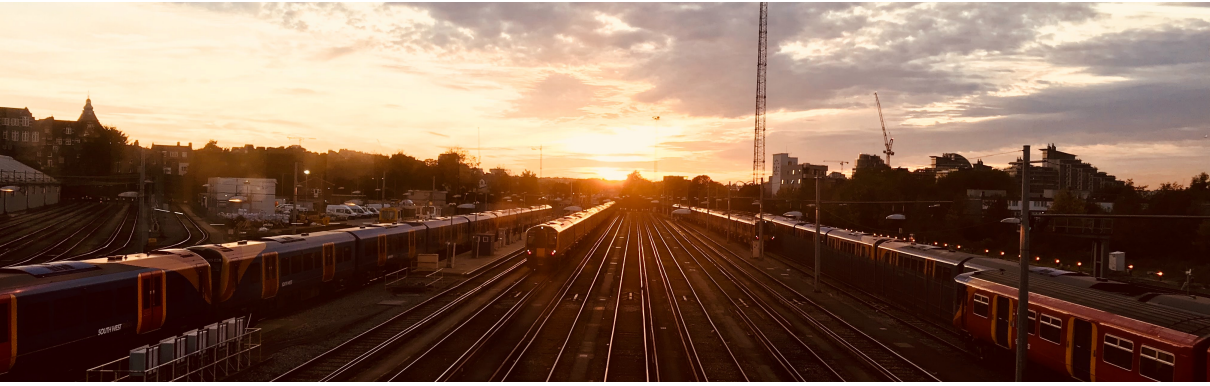

Betriebssysteme SS2024

Koordination & Ausschluss

Tutorium 05

Inhaltsverzeichnis

Nebenläufigkeit

- Nebenläufige Ausführung ist möglich, wenn Anweisungen mehrerer Aktivitätsträger unabhängig voneinander ausführbar sind
- Hierbei müssen Aktivitätsträger nicht aufeinander warten
- Der Scheduler darf nicht für Abhängigkeit zwischen zwei Aktivitätsträger sorgen
- **Problem:** Möglicherweise entstehen Inkonsistenz, wenn zwei nebenläufige Aktivitätsträger auf dieselbe Ressource zugreifen

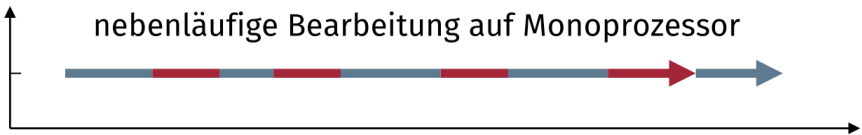

Abbildung: Nebenläufige Ausführung zweier Aktivitätsträger, hier in rot und blau dargestellt

Parallelität

- Gleichzeitige Ausführung der Anweisungen mehrerer Aktivitätsträger
- Parallele Ausführung nur auf Mehrprozessorsystemen möglich
- Die auszuführenden Anweisungen müssen unabhängig voneinander sein
- Parallelität ist Unterkategorie von Nebenläufigkeit

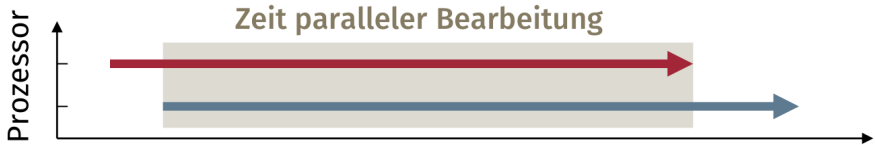

Abbildung: Parallele Ausführung zweier Aktivitätsträger auf einem Mehrkernprozessor

Koordinierung

- Bei der Koordinierung von Prozessen schränkt man die Nebenläufigkeit ein um gleichzeitigen Zugriff auf gemeinsame Ressourcen zu verhindern
- Hierbei wird vor allem die zeitliche Durchmischung der nebenläufigen Aktivitätsträger eingeschränkt damit diese möglichst minimal ist
- Mögliche Maßnahmen:
  - Anhalten eines Prozesses wenn auf gemeinsame Ressource zugegriffen wird
  - Verhindern des Umschaltens durch den Scheduler

Gegenseitiger Ausschluss

- Gegenseitiger Ausschluss ist eine Form der Koordinierung
- Hierbei darf immer nur ein Aktivitätsträger einen kritischen Abschnitt eines Programms betreten
- Kritische Abschnitte werden als Befehlsfolgen angesehen die nicht unterbrochen werden können

Beispiel

Gegeben sei ein Server der mit drei Threads arbeitet. Der erste Thread nimmt Anfragen von Clients entgegen, priorisiert diese, und sortiert diese je nach Priorität in eine Warteschlange ein. Der zweite Thread bearbeitet nun jeweils die vorderste Anfrage in der Warteschlange, erzeugt eine entsprechende Antwort, und übergibt diese an den dritten Thread. Anschließend entfernt er diese Anfrage aus der Warteschlange. Der dritte Thread versendet die ihm übergebenen Antworten. Während des Versendens einer Antwort kann der Thread keine weiteren Antworten entgegen nehmen.

Frage 1

Gegeben sei ein Server der mit drei Threads arbeitet. Der erste Thread nimmt Anfragen von Clients entgegen, priorisiert diese, und sortiert diese je nach Priorität in eine Warteschlange ein. Der zweite Thread bearbeitet nun jeweils die vorderste Anfrage in der Warteschlange, erzeugt eine entsprechende Antwort, und übergibt diese an den dritten Thread. Anschließend entfernt er diese Anfrage aus der Warteschlange. Der dritte Thread versendet die ihm übergebenen Antworten. Während des Versendens einer Antwort kann der Thread keine weiteren Antworten entgegen nehmen.

Frage: Welche der gegebenen Threads können (durchgängig) nebenläufig zueinander ausgeführt werden, welche nicht?

Frage 1

Gegeben sei ein Server der mit drei Threads arbeitet. Der erste Thread nimmt Anfragen von Clients entgegen, priorisiert diese, und sortiert diese je nach Priorität in eine Warteschlange ein. Der zweite Thread bearbeitet nun jeweils die vorderste Anfrage in der Warteschlange, erzeugt eine entsprechende Antwort, und übergibt diese an den dritten Thread. Anschließend entfernt er diese Anfrage aus der Warteschlange. Der dritte Thread versendet die ihm übergebenen Antworten. Während des Versendens einer Antwort kann der Thread keine weiteren Antworten entgegen nehmen.

Frage: Welche der gegebenen Threads können (durchgängig) nebenläufig zueinander ausgeführt werden, welche nicht?

Antwort: Thread 1 ist sowohl von Thread 2, als auch von Thread 3 unabhängig, und kann deswegen nebenläufig mit einem der beiden Threads ausgeführt werden. Thread 2 und 3 sind voneinander abhängig, da Thread 2 ggf. darauf warten muss, dass Thread 3 das Versenden einer Antwort abschließt. Daher können Thread 2 und 3 nicht (durchgängig) nebenläufig ausgeführt werden

Frage 2

Gegeben sei ein Server der mit drei Threads arbeitet. Der erste Thread nimmt Anfragen von Clients entgegen, priorisiert diese, und sortiert diese je nach Priorität in eine Warteschlange ein. Der zweite Thread bearbeitet nun jeweils die vorderste Anfrage in der Warteschlange, erzeugt eine entsprechende Antwort, und übergibt diese an den dritten Thread. Anschließend entfernt er diese Anfrage aus der Warteschlange. Der dritte Thread versendet die ihm übergebenen Antworten. Während des Versendens einer Antwort kann der Thread keine weiteren Antworten entgegen nehmen.

Frage: Können die nebenläufigen Threads auch echt parallel ausgeführt werden?

Frage 2

Gegeben sei ein Server der mit drei Threads arbeitet. Der erste Thread nimmt Anfragen von Clients entgegen, priorisiert diese, und sortiert diese je nach Priorität in eine Warteschlange ein. Der zweite Thread bearbeitet nun jeweils die vorderste Anfrage in der Warteschlange, erzeugt eine entsprechende Antwort, und übergibt diese an den dritten Thread. Anschließend entfernt er diese Anfrage aus der Warteschlange. Der dritte Thread versendet die ihm übergebenen Antworten. Während des Versendens einer Antwort kann der Thread keine weiteren Antworten entgegen nehmen.

Frage: Können die nebenläufigen Threads auch echt parallel ausgeführt werden?

Antwort: Ja, wenn der Server mehrere Prozessorkerne zur Verfügung hat.

Frage 3

Gegeben sei ein Server der mit drei Threads arbeitet. Der erste Thread nimmt Anfragen von Clients entgegen, priorisiert diese, und sortiert diese je nach Priorität in eine Warteschlange ein. Der zweite Thread bearbeitet nun jeweils die vorderste Anfrage in der Warteschlange, erzeugt eine entsprechende Antwort, und übergibt diese an den dritten Thread. Anschließend entfernt er diese Anfrage aus der Warteschlange. Der dritte Thread versendet die ihm übergebenen Antworten. Während des Versendens einer Antwort kann der Thread keine weiteren Antworten entgegen nehmen.

Frage: Welche Probleme können hier durch Nebenläufigkeit auftreten?

Frage 3

Gegeben sei ein Server der mit drei Threads arbeitet. Der erste Thread nimmt Anfragen von Clients entgegen, priorisiert diese, und sortiert diese je nach Priorität in eine Warteschlange ein. Der zweite Thread bearbeitet nun jeweils die vorderste Anfrage in der Warteschlange, erzeugt eine entsprechende Antwort, und übergibt diese an den dritten Thread. Anschließend entfernt er diese Anfrage aus der Warteschlange. Der dritte Thread versendet die ihm übergebenen Antworten. Während des Versendens einer Antwort kann der Thread keine weiteren Antworten entgegen nehmen.

Frage: Welche Probleme können hier durch Nebenläufigkeit auftreten?

Antwort: Da Thread 2 das erste Element aus der Wartenschlange liest, dieses anschließend bearbeitet und danach das erste Element der Wartenschlange löscht, wäre es möglich, dass Thread 1 in dieser Zeit das erste Element ändert. Thread 2 würde dann dieses neu-eingefügte Element löschen und die Anfrage würde verloren geht. Zusätzlich würde die aktuelle Anfrage doppelt verarbeitet werden, da diese als nächstes erneut gelesen werden würde.

Frage 4

Gegeben sei ein Server der mit drei Threads arbeitet. Der erste Thread nimmt Anfragen von Clients entgegen, priorisiert diese, und sortiert diese je nach Priorität in eine Warteschlange ein. Der zweite Thread bearbeitet nun jeweils die vorderste Anfrage in der Warteschlange, erzeugt eine entsprechende Antwort, und übergibt diese an den dritten Thread. Anschließend entfernt er diese Anfrage aus der Warteschlange. Der dritte Thread versendet die ihm übergebenen Antworten. Während des Versendens einer Antwort kann der Thread keine weiteren Antworten entgegen nehmen.

Frage: Wie kann gegenseitiger Ausschluss verwendet werden, um die Probleme der Nebenläufigkeit im Beispiel zu beheben?

Frage 4

Gegeben sei ein Server der mit drei Threads arbeitet. Der erste Thread nimmt Anfragen von Clients entgegen, priorisiert diese, und sortiert diese je nach Priorität in eine Warteschlange ein. Der zweite Thread bearbeitet nun jeweils die vorderste Anfrage in der Warteschlange, erzeugt eine entsprechende Antwort, und übergibt diese an den dritten Thread. Anschließend entfernt er diese Anfrage aus der Warteschlange. Der dritte Thread versendet die ihm übergebenen Antworten. Während des Versendens einer Antwort kann der Thread keine weiteren Antworten entgegen nehmen.

Frage: Wie kann gegenseitiger Ausschluss verwendet werden, um die Probleme der Nebenläufigkeit im Beispiel zu beheben?

Antwort: Die Schreib- und Lesezugriffe auf die Wartenschlange müssen als kritische Abschnitte markiert werden. Bei Thread 2 müssen alle Operationen in einem einzigen kritischen Abschnitt gekapselt werden, um die Konsistenz der Wartenschlange sicherzustellen. Bei Thread 1 genügt es, den Schreibzugriff auf die Wartenschlange als kritischen Abschnitt zu markieren.