Virtuelle Threads

Nur nicht den Faden verlieren!

Was ist ein Thread?

Thread (Informatik)

In der Informatik bezeichnet **Thread** [θɹɛd] (englisch *thread*, ‚Faden', ‚Strang') – auch **Aktivitätsträger** oder **leichtgewichtiger Prozess** genannt – einen *Ausführungsstrang* oder eine *Ausführungsreihenfolge* in der Abarbeitung eines Programms. Ein Thread ist Teil eines Prozesses.

- Prozesse haben einen oder mehrere Ausführungsstränge
- Beispiele:
  - Main-Thread, GC-Threads
  - Rich Client: Event-Thread, UI-Threads, Worker-Threads
  - Servlet-Container: Thread-Pool für Requests, verschiedene Hilfs-Threads
  - Parallele Streams: Thread-Pool

Warum Thread-Pools?

- Pooling: Vorhalten von Ressourcen, deren Bereitstellung teuer ist
- Erzeugung von Betriebssystem-Thread ist teuer
- Thread-Pool üblicherweise im Vorfeld allokiert
- Ein Thread arbeitet nacheinander mehrere Tasks ab
- Problem: Isolation der Tasks

Tasks: Welche Kategorien gibt es?

- Spektrum zwischen zwei Extremen
- CPU-lastig: Task benötigt viel CPU-Zeit, wenig Leerlauf
  - Beispiel: Passwort-Hashing beim Login, Java-Streams
- I/O-lastig: Task benötigt wenig CPU-Zeit, viel Leerlauf
  - Beispiel: Datenbankzugriff, Aufruf von Webservices

Tasks: Rechenbeispiel

- Thread-Pool mit 8 Betriebssystem-Threads
- 24 Tasks
- Jeder Task dauert 10 Sekunden
  - 9 Sekunden warten auf Antwort eines Webservices
  - eine Sekunde CPU-Zeit

- Wie lange dauert die Abarbeitung aller Tasks auf dem Thread-Pool?
- Was wäre, wenn beliebig viele Tasks gleichzeitig warten könnten?

Demo

24 Tasks (9+1 Sekunden), 8 Threads

Optimierung: Mehr Threads

- mehr gleichzeitig wartende Threads
- mehr gleichzeitig laufende Threads
- mehr Kontextwechsel zwischen Threads
- höherer Ressourcenverbrauch

Demo

24 Tasks (9+1 Sekunden), 24 Threads

Optimierung: Reactive Programming

- Publisher-Subscriber
- Pipelines von synchronen und asynchronen Operationen
- Paradigmenwechsel
- schwieriger zu debuggen
- Stacktraces wenig nützlich
- I/O-Schnittstellen müssen auch "reactive" implementiert sein

Optimierung: Koroutinen

- Kotlin: wartende Koroutine blockiert keinen Thread
- verwandtes Konzept in einigen Programmiersprachen: Generatoren
- benötigt geeignete Sprachkonstrukte
- für kooperative Tasks gut geeignet
- in Java nicht einfach umsetzbar

Optimierung: Virtuelle Threads

- JEP 425
- Teil von Project Loom
- in JDK 19 als Preview-Feature verfügbar
- Plattform-Thread vs. virtueller Thread
- basierend auf Continuations

Demo

24 Tasks (9+1 Sekunden), virtuelle Threads

Virtuelle Threads: Eigenschaften

- Virtueller Thread ist ein Java-Objekt ohne Betriebssystem-Äquivalent
- Stack des virtuellen Threads im Java-Heap
- zur Ausführung (CPU) an "Carrier-Thread" gebunden
- kann (und muss) sich selbst vom Carrier-Thread lösen ("unmount")
- später wieder an ggf. anderen Carrier-Thread binden ("mount")
- kein Zugriff auf Carrier-Thread über öffentliche API

Demo

viele Threads

Virtuelle Threads: Kompatibilität

- Instanzen einer Subklasse von java.lang.Thread
- wie gewohnt zu debuggen
- brauchbare Stacktraces
- größtenteils wie "klassische" Threads verwendbar
  - Thread.stop() / .suspend() / .resume() → UnsupportedOperationException
  - Thread.setPriority() → No-Op
  - Thread.setDaemon(false) → IllegalArgumentException
  - nicht von ThreadMXBean unterstützt

Virtuelle Threads: Umdenken

- kein Pooling für virtuelle Threads
  - Millionen virtuelle Threads sind kein Problem
  - ein virtueller Thread pro Task/Request
- keine implizite Beschränkung anderer Ressourcen durch Thread-Pools
  - Beispiel: Webservice erlaubt 10 gleichzeitige Verbindungen
  - explizit z.B. über Semaphoren

Virtuelle Threads: Grenzen

- synchronized-Block bindet virtuellen Thread an Carrier-Thread
  - kann falls notwendig durch ReentrantLock ersetzt werden
  - wird evtl. noch gelöst, bis virtuelle Threads als GA-Feature kommen
- native Methode oder "Foreign Function" (JEP 424) ebenfalls
  - nicht anders möglich, aufgerufener Code außerhalb der JVM
- jstack zeigt virtuelle Threads nicht an
  - jcmd <pid> Thread.dump_to_file -format=json <file>

Demo

synchronized


Fragen?


Vielen Dank!