use forks;
no threads;

Performanter Ersatz für Perls iThreads

Deuscher Perl-Workshop 9.0 / 2007

Alvar C.H. Freude

Threads sind toll!

- Normalerweise sind Threads toll!
- Einfache *und* schnelle Sache, Programmteile parallel laufen zu lassen
- Alle Threads können auf den gleichen Speicher zugreifen
- Man kann eine Aufgabe mal schnell im Hintergrund ablaufen lassen

Threads in Perl sind Mist!

* Perl hat mit den (optionalen) ithreads zwar eine Thread-Implementierung
* Die Syntax ist auch schön und es ist komfortabel (einfach!)
* Aber die iThreads sind eigentlich nur eine Fork-Simulation für Windows und mischen die Nachteile von Forks und Threads

Threads in Perl sind Mist! (2)

* Der Start eines Threads kostet viel Zeit und Speicher: alle Variablen, Module und sonstigen Daten werden *kopiert*
* Alle Variablen und Daten sind nicht geshared, die Modifikation einer globalen Variable in einem Thread hat nur Auswirkung auf den Thread, der sie geändert hat.
* Variablen, auf die alle Threads Zugriff haben sollen, müssen extra als *:shared* deklariert werden. Gesharte Variablen sind sehr langsam (da nicht wirklich geshared)

use forks;

* Mit dem CPAN-Modul *forks* gibt es einen 1:1-Ersatz für ithreads
* Es bietet die gleiche Funktionalität wie die ithreads, nutzt aber intern Fork
* Dadurch muss nicht für jeden Thread der gesamte Perl-Speicher kopiert werden, das macht das Betriebssystem mit Hilfe der MMU
* Dadurch weniger Speicherverbrauch und schneller Start

Nachteile gegenüber ithreads

* Für gesharte Variablen nutzt *forks* TCP- oder UNIX-Sockets.
  * Das ist noch ein wenig langsamer als die Variante der normalen Perl-ithreads.
  * Solange gesharte Variablen aber nur für den Parameteraustausch und (relativ seltene) Statusabfragen und -Meldungen genutzt werden, fällt das nicht auf.

Vorteile gegenüber normalen Forks

- Gegenüber der normalen Verwendung von *fork* gibt es im Wesentlichen zwei Vorteile
  - es ist einfacher zu handhaben
    - und es ermöglicht eine einfache Handhabung von Variablen, auf die alle Prozesse zugreifen können
  - Und dank der Kompatibilität kann man es einfach austauschen

```perl
use forks;

# Startet eine Funktion (berechne_frage) im Hintergrund,
# Übergibt ihr einen Parameter (42) und
# kümmert sich nicht um ihren Rückgabewert

threads->new(\&berechne_frage, 42)->detach;
```

Beispiel 1

Man kann also mit einer einzigen Zeile Code Sachen im Hintergrund berechnen lassen.

```
> perl -Mforks -Mforks::shared app_mit_threads.pl
```

1:1 Ersatz für das threads-Modul

So kann man eine Applikation starten, die ithreads verlangt – aber eben mit forks