THREADING_LOCKS

Abdurrahim Burak Tekin

4AHITM 13.11.2016

Die Aufgabe:

Thread Synchronisation in Python: Summenberechnung

Schreibe ein Programm, welches die Summe von 1 bis zu einer von dem/der Benutzer/in einzugebenden (potentiell sehr großen) Zahl mithilfe von drei Threads berechnet!

Grundanforderungen:

- Eine eigene Klasse erbt von Thread
- Die Klasse definiert eine gemeinsame Lock sowie einen gemeinsamen Counter
- Im Konstruktor wird über einen Parameter bestimmt, für welche Zahlen dieser Thread zuständig ist
- In der run-Methode wird die Summe korrekt aufsummiert, wobei der Zugriff auf den Counter über die Lock threadsicher gestaltet wird (with-Statement)
- Kommentare und Sphinx-Dokumentation
- Kurzes Protokoll über deine Vorgangsweise, Aufwand, Resultate, Beobachtungen, Schwierigkeiten, ... Bitte sauberes Dokument erstellen! (Kopf- und Fußzeile etc.)

Erweiterungen:

- Miss die Laufzeit!
- Untersuche, wie sich die Laufzeit auf deinem System verhält, wenn du es mit mehr oder weniger Threads verwendest, z.B. Single Threaded (d.h. nur im main-Thread), mit 2 Threads, mit 3 Threads, ...
- Interpretiere die Ergebnisse und halte deine Erkenntnisse im Protokoll fest! Warum verhält es sich so?
- Finde eine Möglichkeit, wie die Performance verbessert werden kann und eventuelle Beschränkungen umgangen werden können!

Bemerkungen:

Wichtig ist es Threading zu importieren!

Im Grunde war es bei der Aufgabe fast genauso vorzugehen wie bei der ersten Threading Aufgabe, wo wir einen Code-Encrypter bauen mussten! (mit Threads)

Änderungen gibt es, und zwar Locks: PDF Datei im SEW Moodle-Kurs über Locks ist sehr hilfreich!

Schwierigkeiten:

Schwierigkeiten hatte ich so einige, aber nur das Problematischste war, dass sich am Ende das Ergebnis durch die Anfangszahl subtrahieren ließ!

Ich konnte das Problem lösen indem ich im Parameter zu den Variablen jeweils eine „+1“ dazu gehängt habe!

Warum dieses Problem überhaupt aufgetaucht ist weiß ich leider noch immer nicht!

```
# run-method for counting up
def run(self):
    counter2 = 0
    for i in range(int(self.num), self.thread_num):
        counter2 += i

    with SumThread.lock:
        SumThread.counter += counter2
```

```
What is your chosen number?5
How many Threads do you want?3
================================
Number: 5
Threads: 3
Result: 10
```

Hier existiert das Problem!

```
# run-method for counting up
def run(self):
    counter2 = 0
    for i in range(int(self.num + 1), self.thread_num + 1):
        counter2 += i

    with SumThread.lock:
        SumThread.counter += counter2
```

```
What is your chosen number?5
How many Threads do you want?3
================================
Number: 5
Threads: 3
Result: 15
```

Nach dem Anhängen von „+1“ existiert das Problem nicht mehr!