Aufgabenblatt 6

Aufgabe - Multi-Threading und Synchronisation

Systemkonzepte zum synchronisierten Zugriff sind nicht nur im Zusammenhang mit Betriebssystemen wichtig, sondern auch in der Systemprogrammierung. Sie sollen in dieser Aufgabe ein Programm (einen sog. *Bot*) entwickeln, das aus einer Steuerdatei URLs ausliest. Die Server im Netz sollen kontaktiert und per Http **`GET`** Request die Seiten ausgelesen und lokal gespeichert werden.

Aufgabenstellung:

Eine Steuerdatei listet zeilenweise URLs in der Form:

```
https://www.google.de/
https://www.bing.com/
```

Leerzeilen werden nicht interpretiert. Der Name dieser Steuerdatei kann variiert werden und soll Ihrem Programm über die Kommandozeile als Parameter übergeben werden.

Ihr Programm soll wie folgt arbeiten:

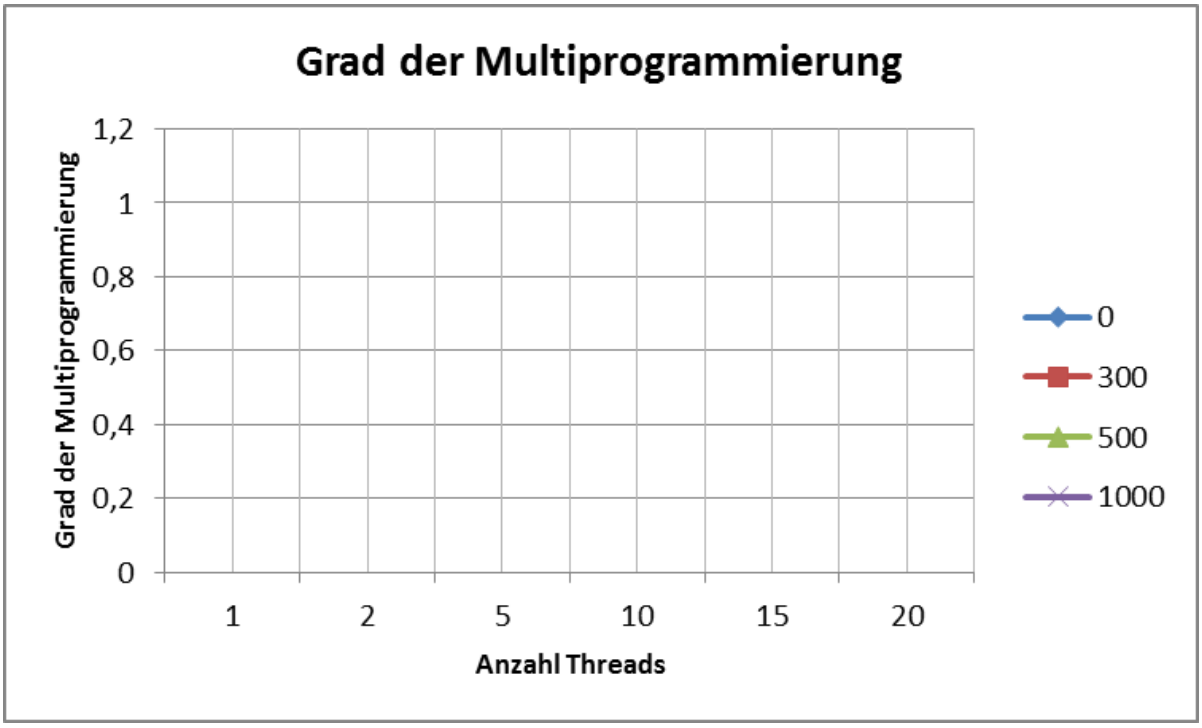
- Das Auslesen der Seiten soll durch **mehrere Threads** geschehen:
  - Ein Thread (*Reader-Thread*) liest die Zeilen aus einer Steuerdatei und schreibt einen entsprechenden Job in eine Queue mit einer (zur Compilezeit) konfigurierbaren Größe.
  - Eine (zur Compilezeit) konfigurierbare Anzahl von Threads (*Client-Threads*) liest die Jobs aus der Queue und kontaktiert die jeweiligen Web-Server.
- Die Zugriffe auf die Queue müssen synchronisiert, d.h. gegeneinander geschützt sein.
- Ein Client-Thread bekommt bei seiner Erstellung als Parameter eine Instanznummer zugeordnet, die ihn z.B. bei Debug-Ausgaben eindeutig identifiziert.
- Die Antworten der Web-Server werden in Dateien gespeichert. Die Dateinamen sollen pro Thread durchnummeriert sein. Sind also n Einträge in der Steuerdatei und wurden m Threads konfiguriert, sollen die Antworten in **`<i>_<j>_<server host>.html`** gespeichert werden (für Dateien mit Index $1 \le i \le n$ und Threads mit Index $1 \le j \le m$).

Bestimmen Sie nun die Laufzeiten des Bots für die Steuerdatei bezüglich unterschiedlicher Konstellationen. Dabei sollen die Zugriffe auf die Seiten um einen einstellbaren Zeitoffset verzögert werden können. Dies ist über im Rahmen des Praktikums mitgelieferte Bibliothek möglich (entsprechende Erläuterungen werden dort gegeben). Messen Sie jeweils die Zeit zur Abarbeitung der URLs in der mitgelieferten Steuerdatei, wobei Sie die Anzahl der Reader-Threads und Zugriffsverzögerung gemäß folgender Tabelle variieren (die Queue-Länge für die Verwaltung der Reader-Threads soll konstant auf 10 eingestellt werden):

	Proxy Verzögerung [ms]			
#Threads	0	300	500	1000
1				
2				
5				

10				
15				
20				

Aus dieser Tabelle kann nun der Grad der Multi-Programmierung bestimmt werden: man bildet dazu für eine feste Verzögerung das Verhältnis der minimalen Laufzeit zur jeweils betrachteten Laufzeit (für die minimale Laufzeit ist der Wert also 100%, sonst kleiner) und trägt dieses Verhältnis über die Anzahl der Threads auf. Zeichnen Sie für die jeweiligen Verzögerungen die Funktionsgraphen. Wie interpretieren diese?



Hinweise:

- Die Funktion zum Zugriff zum Auslesen und Speichern von URLs finden Sie im Verzeichnis **`Web_Request`**. In dem Projekt sehen Sie auch, wie Sie die benötigten Bibliotheken zu dem zu entwickelnden Programm hinzubinden (**`-lm -lssl -lcrypto -pthread`**).
- Mit dem o.g. Projekt wird eine Steuerdatei mit einigen URLs zur Verfügung gestellt, die Sie vor Beginn Ihrer Untersuchungen einmal einzeln abrufen sollten. Es kann nicht ausgeschlossen werden, dass diese temporär nicht zur Verfügung stehen. Modifizieren Sie gegebenenfalls diese Datei im Hinblick auf alternative Server.
- Laufzeiten kann man durch Auslesen der Systemuhr beim Start und beim Ende der Ausführung des Bots messen. Einen Zeitstempel mit Millisekunden-Auflösung [ms] erstellt man per:

```
struct timeval  tv;
gettimeofday(&tv, NULL);
double start = (tv.tv_sec) * 1000 + (tv.tv_usec) / 1000;
```

Für das Testat ist ein Ausdruck in kompakter Form (inkl. Namen der Gruppenmitglieder) des erstellten Programmes und der Auswertung abzugeben.

Testierung: 26.11.2019