Erweiterte Konzepte der Programmierung

Projekt „Webcrawler“

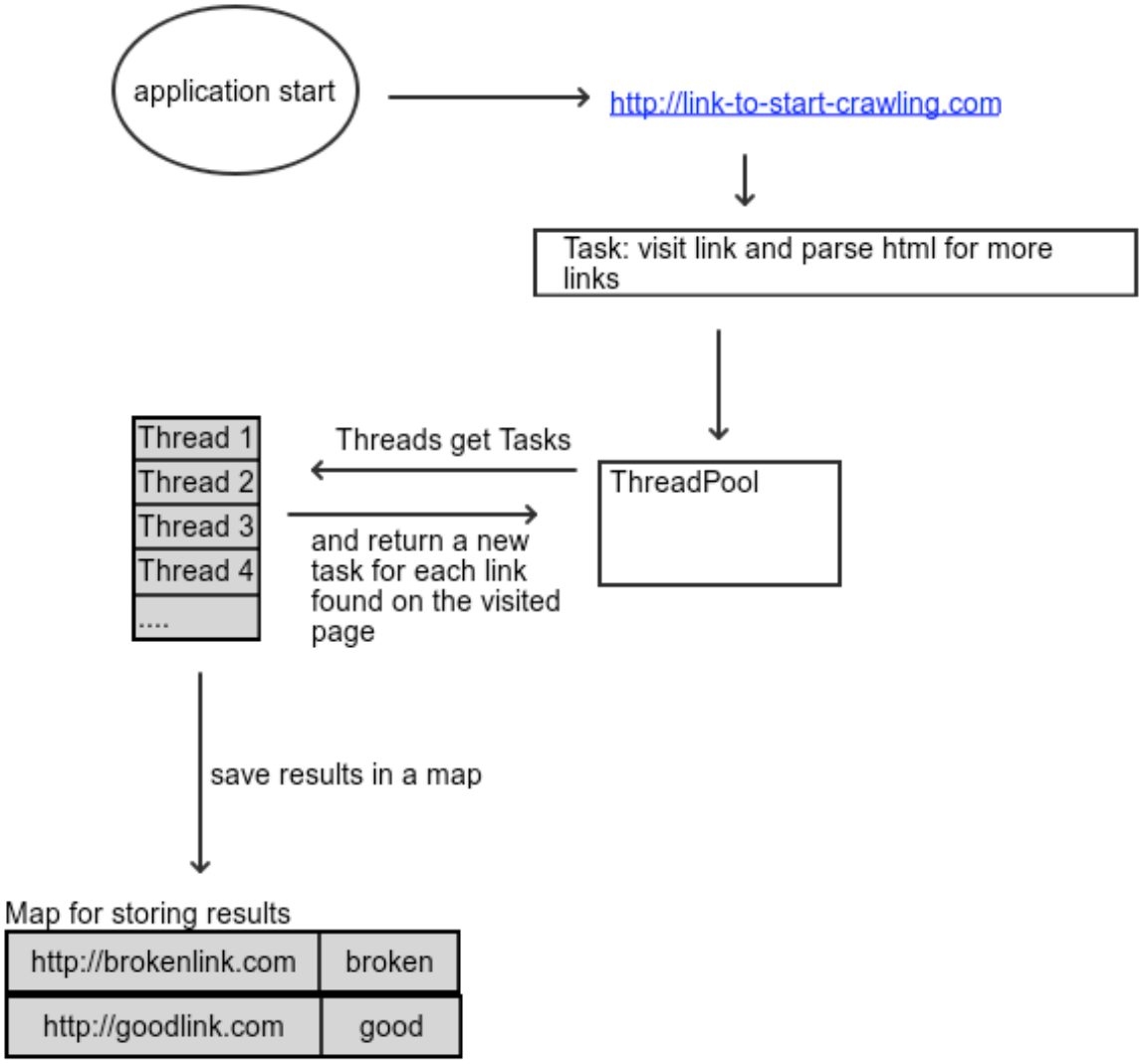
Robert Sommeregger
Dominik Goltermann
Hubert Hölzl

FH-Salzburg, MMT2008

1. **(M) Erzeugen Sie eine sinnvolle Liste von gefundenen broken links und etwas Statistik.**

Statistik über broken links und not-broken links werden in die Datei Log.txt geschrieben.

2. **(M) Beschreiben Sie die Architektur Ihres Programmes. Diagramm!**



3. **(M) Demonstrieren Sie, dass sich Ihr Programm automatisiert sauber bauen lässt. Beschreiben Sie wie's geht.**

Das automatisierte Bauen wurde über ein Makefile gelöst. Im Terminal muss dazu im Ordner "Webcrawler" der Befehl "make" ausgeführt werden.
Installiert sein muss die "Boost-Library" sowie die Library für "htmlcxx".

Der eigentliche Befehl der das Programm kompiliert lautet wie folgt:

```
g++ -g -Wall -Iinclude/ -pthread -o $@ src/main.o src/
ThreadPool.o src/HTTP.o src/HTMLParser.o src/Crawler.o -
lboost_thread-mt -lboost_regex-mt -lcurl –lhtmlcxx
```

4. **(M) Stellen Sie alle Dateien (Code, Dokumentation, Bericht) in einem Repository Ihrer Wahl zur Verfügung**

Als Repository wurde git verwendet, zu finden unter folgender URL:
http://github.com/23tux/busfahrer
Um das Repository zu klonen wird folgender Befehl verwendet:

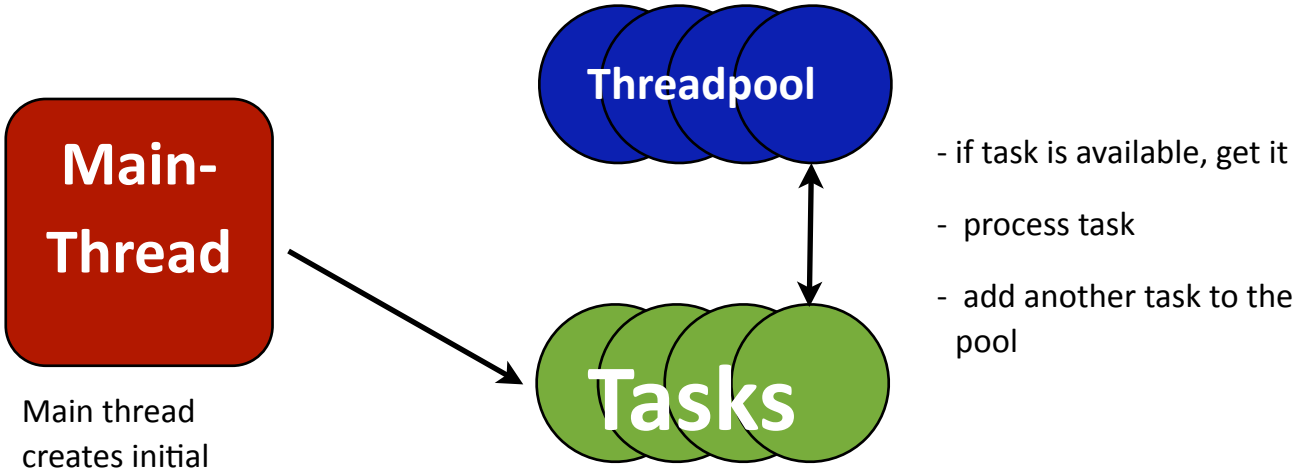
```
git clone git://github.com/23tux/busfahrer.git
```

5. **(M) Beschreiben Sie, welche Teile ihres Programms als Thread ablaufen und wie diese zusammenarbeiten. Diagramm!**

Ein Threadpool der unsere Threads startet nimmt sogenannte Tasks entgegen und beginnt diesen abzuarbeiten. Sobald ein Thread einen Task abgearbeitet hat, wird ein neuer aus einer Taskqueue geholt.

Ein Task selbst besteht dabei aus folgenden Tätigkeiten:

1. Aufrufen einer URL, Bestimmung *"broken / not broken"*
2. Die URL der besuchten Seite wird zusammen mit der *"broken / not broken"* Information in eine Map gespeichert
3. Wenn die URL erreichbar ist, wird der HTML Quellcode geladen und nach Links durchsucht
4. Für jeden gefundene Link wird ein neuer Task in die Taskqueue gestellt



Main thread creates initial task on startup with a given URL

Threads get tasks until there is no one left, then they are waiting for beeing signaled through an submitTask-Event.

If such an event occurs a single thread gets started. Every thread processes one task. A Task is for example a worker class which scans a certain URL, filters another URL on that page and submit further tasks to the threadpool

Optionale eingebaute Funktionen

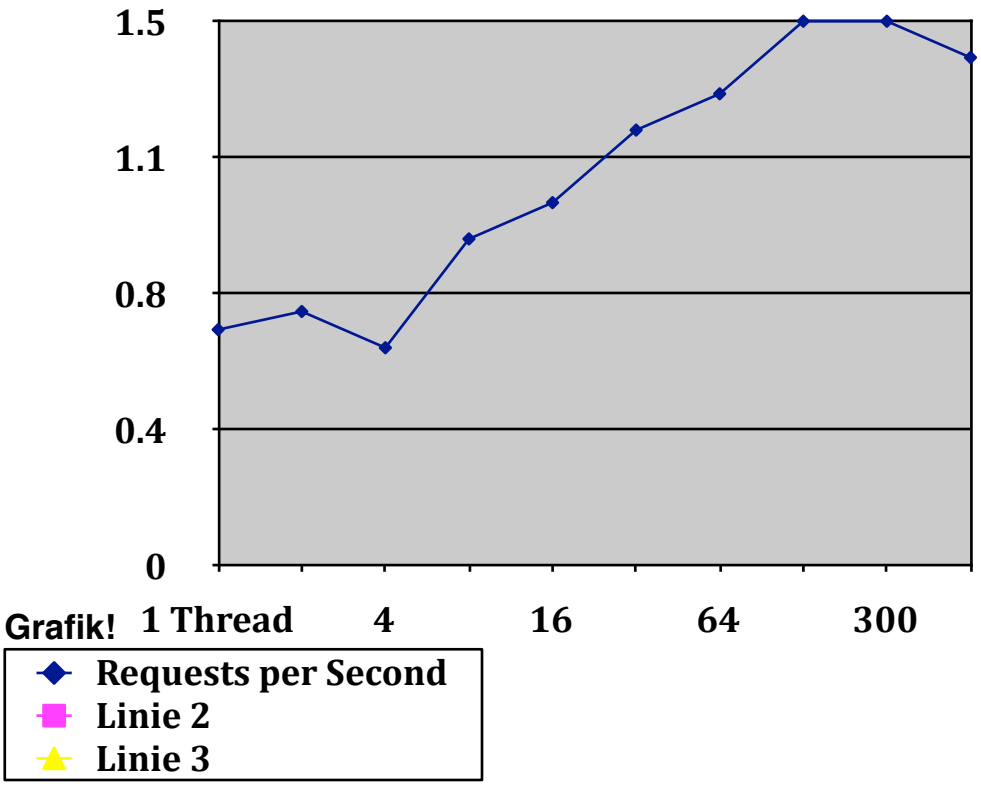
- Ihr Programm soll eine Seite nur einmal untersuchen.
- Ihr Programm kann mit Redirects umgehen?
- Erweitern Sie Ihr Programm so, dass innerhalb einer domain gesucht wird.
- Unser Programm kann mit SSL umgehen
- Linux/Max und Windows Kompatibilität (auf allen Systemen kompiliert) Unser Team entwickelte dieses Programm auf allen drei Plattformen

Führen Sie an, welche Bibliotheken Sie verwenden und unter welcher Lizenz diese stehen.

- HTMLCXX → LGPL
- BOOST → Boost Software License - Version 1.0:

- Curl ☒ MIT/X derivate (curl license):

Untersuchen Sie, ob Ihr Programm durch Multithreading schneller arbeitet (d.h. mehr Webseiten in gegebener Zeit durchsuchen kann).



Grafik!

Die Rate steigt mit der Zahl der Threads stark an. Dies liegt daran, dass ein Request gerade bei meinem DSL 2000 sehr lange dauert. Diese Arbeit lässt sich also sehr gut synchronisieren. Ab einer gewissen Zahl von Threads (in meinem Fall ca. ab 150 Threads) scheint meine Internetverbindung ausgelastet zu sein und es wird eher langsamer wenn man dann noch mehr Threads benutzt.