

Erweiterte Konzepte der Programmierung

Projekt „Webcrawler“

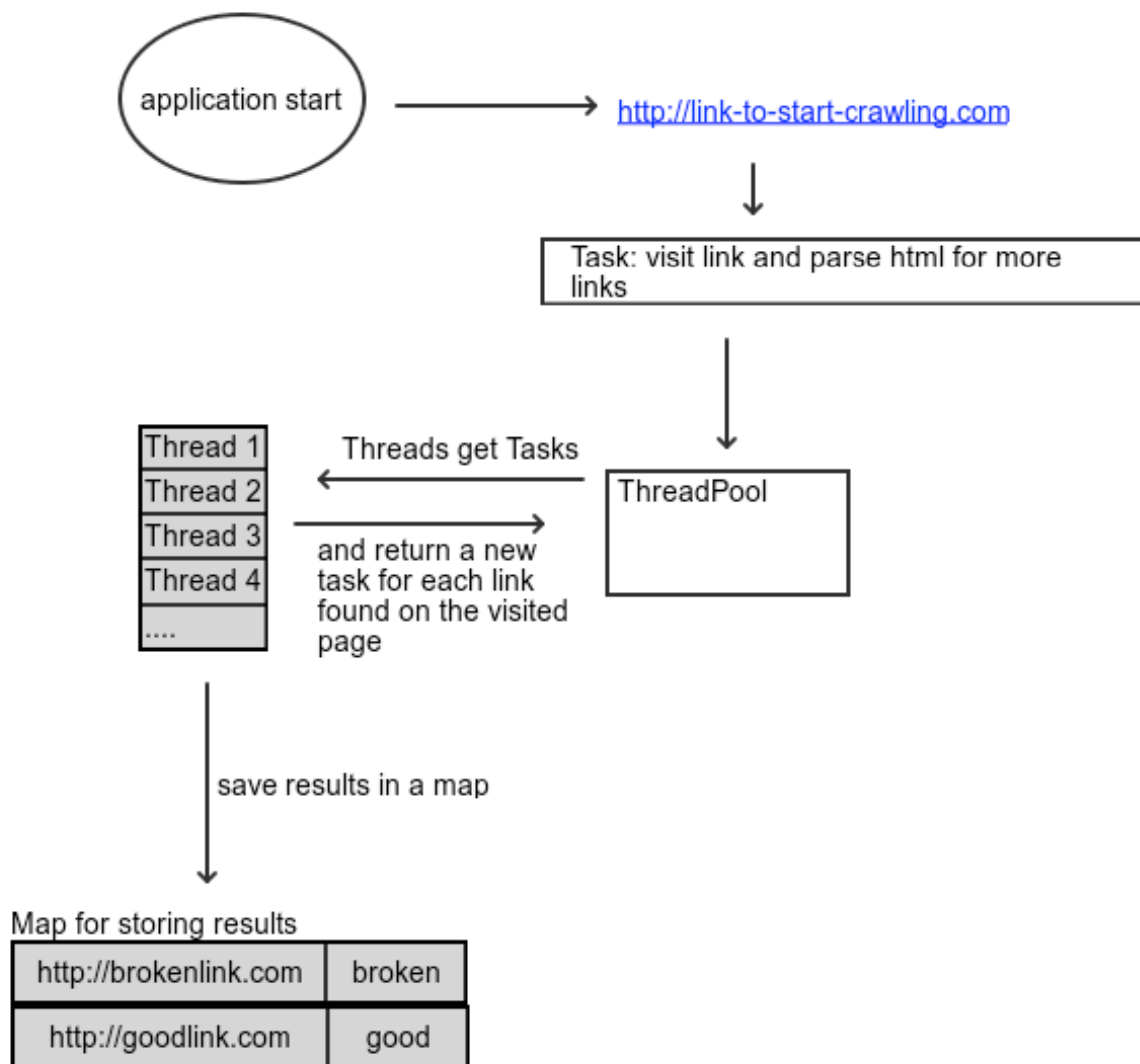
Robert Sommeregger
Dominik Goltermann
Hubert Hölzl

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:

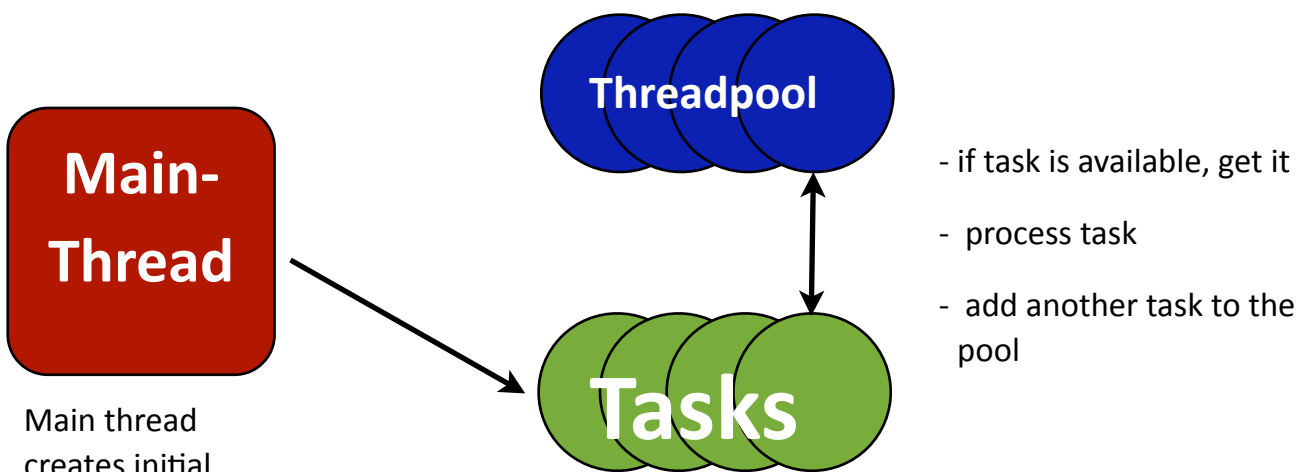
```
git clone git://github.com/23tux/busfahrer.git
```

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

- if task is available, get it
- process task
- add another task to the pool

Threads get tasks until there is no one left, then they are waiting for being 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:

Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:

The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

- Curl  MIT/X derivate (curl license):

COPYRIGHT AND PERMISSION NOTICE

Copyright (c) 1996 - 2010, Daniel Stenberg, <daniel@haxx.se>.

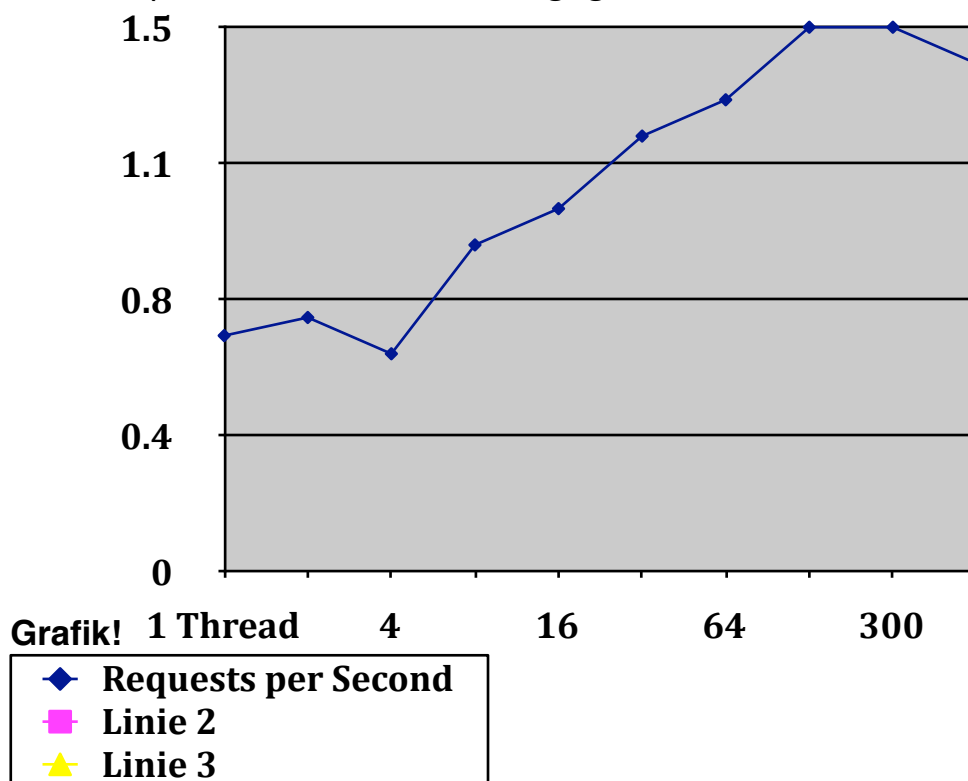
All rights reserved.

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

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



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.