Wie viele Schiffe fahren an der Hochschule vorbei?

Hochschule Bremerhaven

Das STEP-Projekt von Team04 in 2023

Anette Michlik, Kasem Rashrash, Kilian Schramm, Marc Rabus und Paula Urban

Einleitung

STEP ist das Modul, welches den Studienstart mittels eines Projekts für alle Studienanfänger/-innen erleichtern soll und die wesentlichen Werkzeuge einer UNIX-Umgebung vermittelt, die zur Verarbeitung von Datenströmen benötigt werden. Im letzten Drittel des Semesters sucht sich jedes Team eine Fragestellung heraus, um mit einem gegebenen Datenstrom eine dynamische Webanwendung zu eben dieser Fragestellung umzusetzen. In unserem Fall sammeln wir AIS-Daten, die von Schiffen in einem 20 Kilometer-Radius um die Hochschule herum gesendet werden, um anschließend die Anzahl der Schiffe daraus zu entnehmen. Neben diesem Abschlussprojekt gibt STEP auch eine Einführung in wissenschaftliches Arbeiten, indem ein wissenschaftlicher Artikel analysiert wird, sowie in das Arbeitsumfeld von (Wirtschafts-)Informatiker/-innen durch eintägige Unternehmensbesuche.

Um unsere Fragestellung zu beantworten, galt es, die wesentlichen Arbeitsschritte zu erkennen und sinnvoll in unserem Team aufzuteilen. Hier zeichnen sich drei große Blöcke ab:
- Die Daten sicher abfangen und in eine Datei schreiben
- Aus den Daten herauslesen, wie viele Meldungen in der Stunde eingehen, um daraus einen Gnuplot zu generieren
- Einen Lasttest für die Anwendung durchführen

Diese Skripte haben wir untereinander aufgeteilt und für statische Schiffdaten mit zuvor bereits gesammelten Daten in unseren eigenen Verzeichnissen getestet. Fertige Skripte oder Teile davon, haben wir in unserem Gruppen-User gesammelt, besprochen und eventuell erweitert. Wichtig hier zu erwähnen ist, dass wir nie auf den echten Dateien im Gruppen-User gearbeitet haben, sondern uns immer eine Kopie der Skripte gemacht haben. Hintergrund ist, dass so nicht zwei Menschen auf einer Datei arbeiten können und beim Speichern die Änderungen des Anderen verloren gehen. Nach einer kurzen Teamabsprache haben wir die Skripte dann endgültig aktualisiert. Erst im vorletzten Schritt haben wir unsere Anwendung mit dem Bau einer html-Seite dynamisch gemacht und in unseren Docker-Container geschoben. Der letzte Schritt war dann das Durchführen und Aufbereiten von Lasttests.

Prototyp

An dieser Stelle soll Genaueres zur Umsetzung des Prototypen erläutert werden. Hierbei ist noch wichtig zu erwähnen, dass wir die Daten nicht in einer großen Datei speichern, sondern pro Tag, an dem die Anwendung läuft in je eine Datei. Das hat den Vorteil, dass die Dateien nicht unkontrollierbar groß werden und es sich gut darüber iterieren lässt.

(Watcher)-Skript

Für STEP steht die Funktionalität der Anwendung im Vordergrund. Diese hängt an der Sicherstellung eines laufenden Datenstroms zu Rhodes, dem Empfänger der AIS-Daten, weshalb wir dies durch ein eigenes (Watcher-)Skript gewährleisten. Dieses Skript wird mit einem Cronjob genau einmal in der Minute gestartet, und prüft, ob der Datenstrom weiterhin läuft. Für das Skript selbst unterscheiden wir konzeptionel zwei Fälle.
Fall 1 ist, dass es einen Verbindungsabbruch zu Rhodes gab, so muss überprüft werden, ob in der Liste der laufenden Prozesse ein ncat zu Rhodes besteht und ob dieser gegebenfalls neu gestartet werden muss. Es kann nun passieren, dass unser User eine Verbindung zu Rhodes hat, hier aber keine Daten mehr an uns gesendet werden. Dieser Fall 2 wird erkannt, wenn der letzte Datensatz, der von Rhodes gesendet ist, länger als eine Minute her ist. Hier wird der Prozess, nämlich die Verbindung zu Rhodes, den wir zuvor in einer Datei mit der ID gespeichert haben, gestoppt, und ein neuer Prozess zur Verbindung mit Rhodes erstellt.

Wie wir die Frage beantworten

Der zweite Teil unserer Anwendung beschäftigt sich mit unserer Fragestellung, wie viele Schiffe pro Tag im Umkreis der Hochschule AIS-Daten senden. Um eben das zu visualisieren, generieren wir pro Datumsanfrage eine Gnuplotgrafik. Das Kozept dahinter zeigt das anschließende UML-Diagramm.

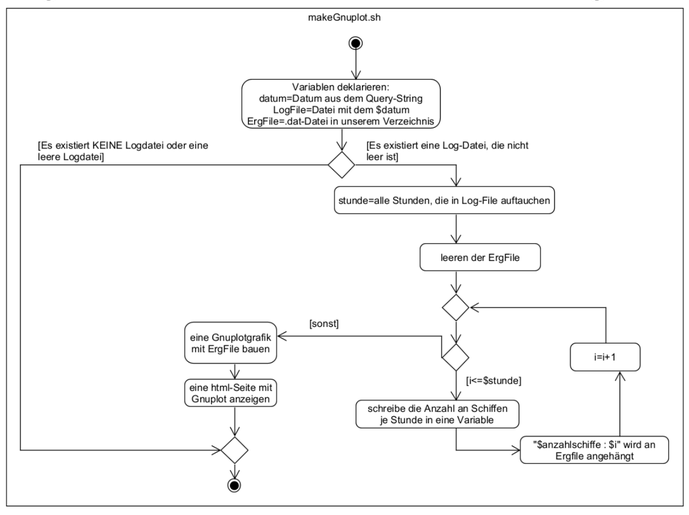

Abbildung 1: Aktivitätsdiagramm zur Erzeugung unserer Gnuplotgrafik

An dieser Stelle sei noch auf eine Besonderheit der Anwendung hingewiesen. Denn durch das Leeren der .dat-Datei, bevor wir in eben diese die angefragten Schiffsdaten eintragen, kommt unsere Anwendung mit genau einem Gnuplot aus. Wir generieren also nicht pro Anfrage einen eigenen Gnuplot, sondern überschreiben diesen immer wieder. Im Code wird dies durch das einfache Umlenken in eine Datei “>“ implementiert. Dieses Vorgehen gehört durch das Verwenden einer “grüneren Logik mit optimierter Programmierung“ zum Kozept des Green Codings, das nachhaltige Softwareentwicklung beschreibt.[1, S.11]

```
1  #!/usr/bin/env bash
2  LogFile="/usr/lib/cgi-bin/rhodes/logs/rho
   ↪  des-$datum.log""
3  ErgFile="/usr/lib/cgi-bin/rhodes-$datum/a
   ↪  nzahl-pro-stunde.dat"
4  > $ErgFile
5  for i in $stunde; do
6  anzahlSchiffe=$(grep "$i:" "$LogFile"|
   ↪  cut -d '|' -f 1 | sort | uniq)
7  echo "$i:00 $mmsi" >> "$ErgFile"
8  done
```

Listing 1: makegnuplot.sh

Der Lasttest

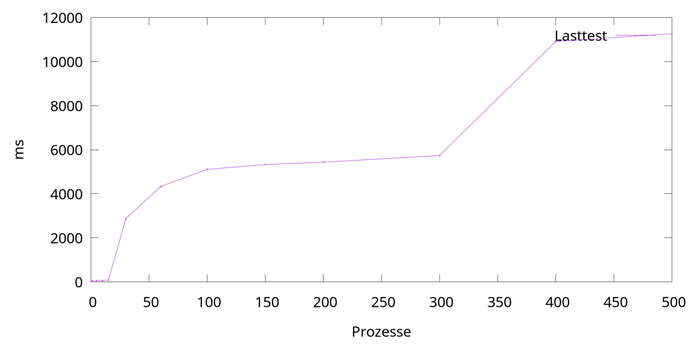

Abbildung 2: wie reagiert unsere Anwendung auf viele Nutzer gleichzeitig

Der Lasttest dient dazu, herauszufinden, wie oft unsere Anwendung gleichzeitig gestartet werden kann, ohne dass es zu einer Verlängerung der Antwortzeit für den Anwender im Browser kommt. Wir beobachten, dass es ungefähr 40 gleichzeitige Prozesse braucht, um unsere Anwendung über die “magischen“ zwei Sekunden zu bringen.

Fazit

Das Ergebnis unserer Anwendung ist, dass pro Stunde zwischen 70 und 80 Schiffe Daten senden, was in Anbetracht der Nähe zum Industriehaven Bremen nicht allzu unrealistisch erscheint. Daneben ist ein viel größeres Ergebnis von STEP, das auch im ersten Semester in Teams schon kleine Anwedungen im Browser umgesetzt werden können.

Referenzen

[1] S. Winkler, J. Günther und R. Pfenning, „Nachhaltige Digitalisierung oder Nachhaltigkeit durch Digitalisierung?“ *HMD Praxis der Wirtschaftsinformatik*, Jg. 60, Nr. 4, 2023, ISSN: 2198-2775. DOI: 10.1365/s40702-023-00987-9.