Programmeertechnieken 2016
Opdracht 1: Data Analysis Pipelines

Deadline: Vrijdag 19 februari, voor het einde van de dag.

1 Introductie

Het analyseren van datasets is een veelvoorkomende taak in de Informatica. Er zijn vele verschillende typen datasets, zoals bijvoorbeeld een dataset met resultaten van een set experimenten, "access logs" van een webserver of voorraadbeheer van een bedrijf opgeslagen in een relationeel databasesysteem. Op een dataset kunnen analyses worden uitgevoerd met variërende complexiteit: eenvoudige analyses zoals het tellen hoe vaak een bepaalde karakteristiek voorkomt (hoe vaak is een bepaalde pagina van een website bezocht in de laatste maand?) tot ingewikkelde analyses waarbij er bijvoorbeeld wordt gezocht naar patronen in de data (als iemand product A koopt, wat voor ander type producten zitten er dan in dezelfde aankoop?). In deze opdracht beperken we ons tot de eenvoudigere analyses. Het zoeken naar patronen in data komt later in de studie uitgebreid aan de orde bij het vak Data Mining.

In deze opdracht gaan we "access logs" van webservers analyseren. In deze logbestanden wordt bijgehouden welke pagina's van een website worden opgevraagd, wanneer, vanaf welk IP-adres en door welke webbrowser. De opdracht bestaat uit het genereren van verschillende statistieken gegeven een set van access logs. Bij het uitvoeren van de analyses is het de bedoeling optimaal gebruik te maken van de kracht van UNIX-systemen: in plaats van zelf één groot programma te schrijven dat de analyse uitvoert, maken we een aaneenschakeling van al bestaande tools op een UNIX systeem. In een dergelijke aaneenschakeling wordt de uitvoer van het ene programma steeds doorgestuurd naar een ander programma. Het doorsturen van data wordt gedaan met behulp van een "pipe", vandaar dat voor dergelijke aaneenschakelingen vaak de term "pipeline" wordt gebruikt. Informatie over standaard tools kan worden opgezocht met het commando *man*: met `man -k <zoekterm>` kan er worden gezocht naar de naam van een programma, een gedetailleerde handleiding kan worden opgevraagd met `man <programmanaam>`.

Het komt wel eens voor dat er nog geen geschikt programma bestaat om in de pipeline op te nemen. In dat geval zit er niets anders op dan het programma zelf te schrijven. Vaak wordt er gebruik gemaakt van een scripttaal, zoals Perl of Python, om in weinig tijd een geschikt programma te schrijven. We zullen dat in deze opdracht ook doen, in ieder geval bij het opzoeken van het land waarin een IP-adres zich bevindt en voor het maken van grafieken op basis van de data. Houd bij het ontwikkelen van deze extra programma's de UNIX filosofie in het achterhoofd! Maak de programma's zo generiek mogelijk, zodat je ze in de toekomst zou kunnen hergebruiken.

2 Dataset

Als dataset zullen we gebruik maken van een uittreksel van 25 dagen uit de access logs van de website van de World Cup 1998[1]. Het betreft hier een publieke dataset. Omdat in de publieke dataset de IP-adressen zijn verwijderd omwille van privacy, hebben we fictieve IP-adressen teruggeplaatst in de bestanden. De databestanden zijn klaar gezet op de "NUWD" computers, zie de appendix. Ongecomprimeerd omvat de database ongeveer 4 GB aan data (46 miljoen regels). Je kunt je programma's testen met losse bestanden. Voor het verslag is het de bedoeling dat *alle* data wordt verwerkt.

De bestanden zijn tekstbestanden geformatteerd als "Apache access log". Elke regel in het bestand is één enkel verzoek aan de webserver. Je vindt onder andere de volgende velden terug, gescheiden door spaties:

- IP-adres waar het verzoek vandaan kwam.
- Timestamp wanneer het verzoek binnen kwam.

[1] `http://ita.ee.lbl.gov/html/contrib/WorldCup.html`

- Het exacte HTTP verzoek, tussen aanhalingstekens.
- De HTTP reply code[2].
- Aantal bytes dat is verstuurd.

3 Te verrichten analyses

De opdracht is nu als volgt: bouw geschikte pipelines om de volgende vier statistieken te genereren:

1. Een lijst van alle HTTP reply codes die voorkomen waarbij het verzoek niet `OK` (code 200) is.

   (Kijk bijvoorbeeld naar `grep -v` *en wellicht komt* `awk` *van pas?).*

2. Overzicht top tien landen van waaruit de meeste bezoekers kwamen, gesorteerd op aantal bezoeken.

   Pak dit bijvoorbeeld als volgt aan:

   - *Lees het gecomprimeerde bestand van de disk (bijv.* `bzcat`*).*
   - *Isoleer het IP-adres aan het begin van elke regel (bijv.* `cut`*).*
   - *Schrijf een Python script dat gebruik maakt van de "GeoIP" module om elk IP-adres om te zetten in een landcode.*
   - *Sorteren en tellen (bijv.* `sort`*,* `uniq -c`*).*
   - *Laat alleen de bovenste 10 regels zien (bijv.* `head`*).*

3. Een grafiek van het aantal bytes dat er gemiddeld per uur van de dag wordt verzonden.

   (Schrijf bijvoorbeeld een script dat gegeven een timestamp deze om kan zetten naar een unieke dag en het uur van de dag (kijk eens naar de `datetime` *module). Dan staat alle data klaar om te sommeren en gemiddelden te berekenen. Tenslotte zou je een Python script kunnen schrijven dat gegeven een lijstje van uren van de dag en gemiddeld aantal verzonden bytes voor dat uur een PDF-bestand met een grafiek genereert (bijvoorbeeld* NumPy *en* matplotlib *of* pandas*, zie ook de appendix)).*

4. Een boom die weergeeft hoe vaak elke directory van de website is bezocht.

   Denk hierbij aan het volgende:

   - *Beschouw alleen de directory van elk bezoek, dus haal de bestandsnaam uit het pad.*
   - *Beschouw alleen succesvolle en "redirection" log entries.*
   - *Je kan weer gebruik maken van* `uniq -c` *om te tellen.*
   - *Schrijf een script dat* `uniq -c` *uitvoer accepteert en hier een ASCII boom voor genereert. Er mag gebruik worden gemaakt van bestaande Python modules om de boom te genereren, bijvoorbeeld* `asciitree`*.*

4 Vereisten

Er mag worden gewerkt in teams van twee personen. Het volgende moet worden ingeleverd:

- De source code van alle geschreven scripts.
- Shell scripts welke de geschreven pipelines uitvoeren.

[2] Zie ook `https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html`.

- Een kort verslag in PDF formaat, gemaakt met behulp van LaTeX, waarin per analyse wordt uitgelegd hoe de pipeline is opgebouwd, de uitvoer van de pipeline wordt getoond (grafieken, tabellen of tekst) en de genereerde statistieken kort worden bediscussieerd. Voeg de gemaakte tabellen/grafieken dus in je verslag! De code hoef je *niet* in het verslag te zetten.
- Let op: alle 25 databestanden moeten worden meegenomen in de analyse.

Zorg ervoor dat alle bestanden die worden ingeleverd (source code, verslag, enz.) zijn voorzien van naam en studentnummer! Plaats alle bestanden om in te leveren in een aparte directory (bijv., `opdracht1`) en maak een "gzipped tar" bestand:

```
tar -czvf opdracht1-sXXXXXXX-sYYYYYYY.tar.gz opdracht1/
```

Vul op de plek van `XXXXXXX` en `YYYYYYY` de bijbehorende studentnummers in. De inzendingen kunnen worden verzonden per e-mail naar `pt2016 (at) handin.liacs.nl` met als onderwerp "PT Opdracht 1". Het verslag hoeft *niet* op papier te worden ingeleverd, zoals bij Programmeermethoden het geval was.

Normering: Verslag (3/10), werking (4/10), kwaliteit code/commentaar/modulariteit (3/10).

5 Bonuspunt

Er kan een bonuspunt worden verdiend door de pipelines zo in te richten dat multi-core processoren optimaal worden benut. Hierbij kan er bijvoorbeeld gebruik worden gemaakt van "GNU parallel"[3], welke klaar is gezet op de NUWD computers (zie appendix voor de benodigde environment setting). Om in aanmerking te komen voor het bonuspunt moet er in het verslag worden omschreven hoe de data analyse precies is geparallelliseerd.

Appendix: Tips & Tricks

Databestanden

De databestanden zijn klaargezet op de NUWD systemen en zijn te vinden in de volgende directory:

```
/vol/share/groups/liacs/scratch/pt2016/opdracht1/
```

GeoIP Python module

De GeoIP Python module is in een speciale directory geïnstalleerd. Alvorens deze module te gebruiken, voer in de terminal dan eerst het volgende commando uit (welke een aantal environment variabelen, zoals `PATH`, goed instelt):

```
source /vol/share/groups/liacs/scratch/pt2016/pt2016.env
```

Hoe moeten we nu deze module gebruiken? Hieronder volgt een kort voorbeeld. In de Python interpreter kun je na het uitvoeren van het `import` commando, met `help(GeoIP)` meer hulp krijgen over de module.

```
import GeoIP
gi = GeoIP.new(GeoIP.GEOIP_MEMORY_CACHE)
gi.country_code_by_addr("132.229.129.158")
gi.country_code_by_name("www.liacs.nl")
```

[3] `http://www.gnu.org/software/parallel/`

GNU Parallel

GNU Parallel is in dezelfde directory geïnstalleerd als de GeoIP module. Voer hetzelfde `source` commando uit om GNU Parallel te kunnen gebruiken.

Eenvoudig plot voorbeeld

Het volgende voorbeeld leest een reeks regels van *stdin* welke bestaan uit (bijvoorbeeld) het uur van de dag en het aantal verstuurde bytes, gescheiden door een spatie. Gebaseerd op deze data wordt een plot gemaakt. Je kunt dit als startpunt gebruiken. Bekijk de documentatie van `matplotlib` om de plot verder op te maken.

```
import pandas
import matplotlib.pyplot as plt
import sys

D = pandas.read_csv(sys.stdin, sep=" ", header=None, index_col=0)
D.plot(kind="bar", rot=30, legend=False)
plt.title("Mijn plot")
plt.show()
exit(0)
```