СЛЁРМ Обучение для инженеров и технических лидеров

Дата-инженер

Linux-терминал для задач Data Engineering

Николай Марков

Цели урока. Что вы узнаете:

1. Как дата инженеры решают Ad-Hoc задачи
2. Как работать с текстовыми данными
3. Как делать запросы в API из терминала
4. Что такое консольные потоковые редакторы, и как работать с JSON и CSV без всякого Python
5. Как организовать параллельные вычисления в терминале

Ad-hoc задачи для дата-инженеров

Где и как хранятся данные?

- В базах данных наподобие PostgreSQL
- Чуть более специализированные форматы — HDF5
- Различные API
- Файлы на HDFS или в локальных файлах
- Запакованные каким-то архиватором (tar.gz, lzo, xz, zip)
- С датами или таймстемпами в названиях
- Иногда в сложной структуре каталогов
- В формате CSV/TSV (колонки данных) или JSON

Bash

- Есть практически на любой *nix/BSD системе
- Любые операции элементарно автоматизируются созданием скриптов
- Большинство кода УЖЕ написано за нас
- Огромное количество программ имеют CLI либо версию, работающую в окне терминала (rtorrent, midnight commander, мессенджеры, почтовые клиенты, архиваторы, браузеры)

Копирование/перемещение файлов

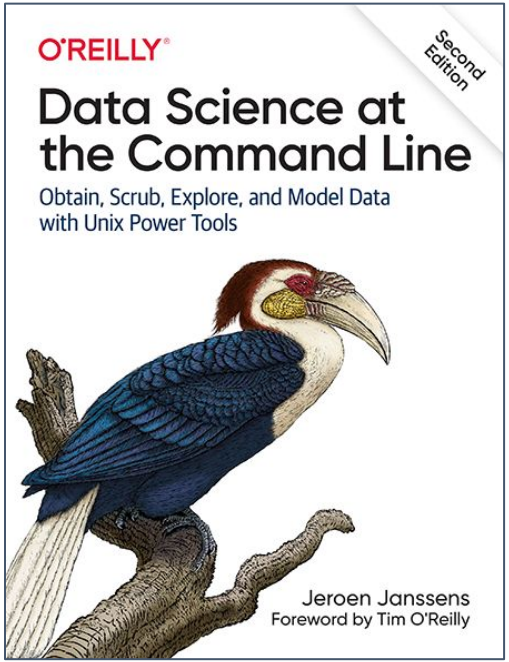
Простые статистические метрики

Извлечение колонок из данных

Распаковка архивов на лету

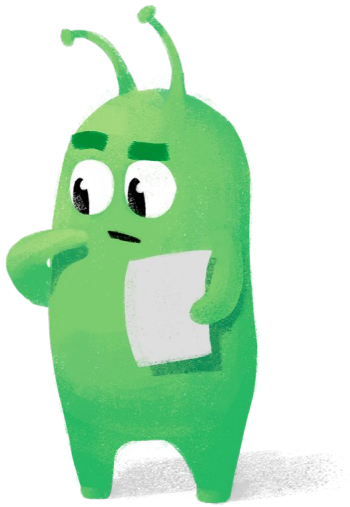
Кэширование

Вдохновение



[Консольные утилиты в 235 раз быстрее Hadoop-кластера]

Итак, у меня есть текстовые данные…



Базовый набор команд

```
cat
wc -l
tar -xvf
sort
echo
ps aux
chmod u+x
ls -la
grep
less
find
```

Первые шаги

~$ seq [first [incr]] last # последовательность чисел
~$ tr ‘first’ ‘second’ # замена символов во входных строках
~$ zcat / gunzip -c # распаковка файла с выводом на терминал
~$ head -n 10 # вывод первых нескольких строк
~$ tail -n 10 # вывод последних нескольких строк
~$ uniq -c # подсчет количества уникальных строк

~$ tail -n+100 # вывод последних строк, начиная со строки 100
~$ zgrep # grep по архивированным файлам
~$ seq -f "Line %g" 10 # последовательность с шаблоном

Кодировка

enca

iconv

Трубоукладка

~$ cat file.csv | grep -v Title | tr ‘n’ ‘0’ | awk ‘{ print $5 }’

Система реализует буферизацию, разделение данных на куски и управление памятью за нас

~$ set -o pipefail *# перехват кодов выхода*

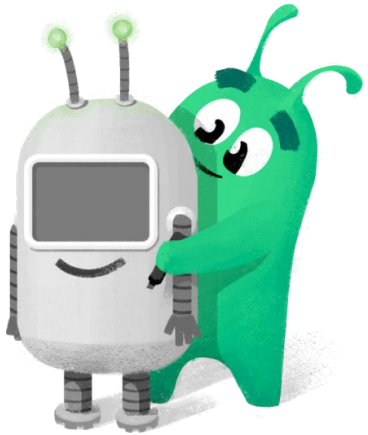
~$ { echo "1"; echo "2"; } | other_command *# объединение вывода*

~$ sleep 3 | echo 1 *# немного магии*
~$ cat file | (grep "foo" || true) | less *# еще немного магии*

https://habrahabr.ru/post/195152/

http://ryanstutorials.net/linuxtutorial/piping.php

Запросы в API из терминала



Web API

~$ curl -s https://www.gutenberg.org/files/76/76-0.txt

~$ curl [-XGET|-XPOST|-XPUT|-XHEAD]

~$ curl -s http://url/some_file | tr '[:upper:]' '[:lower:]' | grep -oE '\w+' | sort | uniq -c | sort -nr -k1,1 | head -n 100

Для надёжности - сохраняйте файл

HTTPie

```
[~] http PUT httpbin.org/put hello=world
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: *
Connection: keep-alive
Content-Length: 452
Content-Type: application/json
Date: Thu, 06 Dec 2018 15:43:38 GMT
Server: gunicorn/19.9.0
Via: 1.1 vegur
```

httpie

[Оригинальный репозиторий]

Потоковые редакторы и форматы

Потоковый редактор sed

~$ cat file.txt | sed ’s/First/Second/‘ # потоковая замена

~$ cat file.txt | sed ’s/First/Second/g‘ # жадная потоковая замена

~$ cat file.txt | sed ’s:/one/path:/another/path:g’ # другой разделитель

~$ cat file.txt | sed ’s/[a-z]*/(&)/g’ # ссылка на ту же строку

~$ cat file.txt | sed ’s/(Foo)[a-z]*/\1fel/g’ # ссылка на ту же строку

~$ sed -r/-E … # расширенные регулярные выражения

http://www.grymoire.com/Unix/Sed.html

Потоковый редактор sed

awk круче cut

~$ cat file.txt | awk -F':' '{ print $2 }' # вывод поля по номеру

~$ cat file.txt | awk '{ print $2 "," $1 }' # join двух полей запятой

~$ cat file.txt | awk 'BEGIN { OFS="\t" }{$1=$1; print $1,$2 }' # меняем разделитель

~$ cat file.txt | awk '{ x+=$2 } END { print x }' # суммируем поле 2

~$ cat file.txt | awk -v home=$HOME/project '{ print home "/data.txt" }'

http://www.grymoire.com/Unix/Awk.html

Переменные:
NF — число полей
NR — число строк

Работа с CSV

csvkit — модуль Python для продвинутой работы с CSV
~$ pip install csvkit

in2csv, csvcut, csvlook, csvjson, csvsql, csvsort

~$ in2csv imdb-250-1996-2011-lists-only.xlsx 2>/dev/null | csvsql --query "select Title,Year from stdin where Year<2009" | csvsort -r -c Year | head -n 10 | csvlook

https://csvkit.readthedocs.org/

Работа с CSV побыстрее

```
$ cat example.csv
color,shape,flag,index,quantity,rate
yellow,triangle,1,11,43.6498,9.8870
red,square,1,15,79.2778,0.0130
red,circle,1,16,13.8103,2.9010
red,square,0,48,77.5542,7.4670
purple,triangle,0,51,81.2290,8.5910
red,square,0,64,77.1991,9.5310
purple,triangle,0,65,80.1405,5.8240
yellow,circle,1,73,63.9785,4.2370
yellow,circle,1,87,63.5058,8.3350
purple,square,0,91,72.3735,8.2430
```

```
$ mlr --icsv --opprint sort -f color,shape example.csv
color  shape    flag index quantity rate
purple square   0    91    72.3735  8.2430
purple triangle 0    51    81.2290  8.5910
purple triangle 0    65    80.1405  5.8240
red    circle   1    16    13.8103  2.9010
red    square   1    15    79.2778  0.0130
red    square   0    48    77.5542  7.4670
red    square   0    64    77.1991  9.5310
yellow circle   1    73    63.9785  4.2370
yellow circle   1    87    63.5058  8.3350
yellow triangle 1    11    43.6498  9.8870
```

```
$ mlr --icsv --ojson filter '$color=="yellow"' example.csv
{
  "color": "yellow",
  "shape": "triangle",
  "flag": 1,
  "index": 11,
  "quantity": 43.6498,
  "rate": 9.8870
}
{
  "color": "yellow",
  "shape": "circle",
  "flag": 1,
  "index": 73,
  "quantity": 63.9785,
  "rate": 4.2370
}
{
  "color": "yellow",
  "shape": "circle",
  "flag": 1,
  "index": 87,
  "quantity": 63.5058,
  "rate": 8.3350
}
```

```
$ mlr --c2p --from example.csv put '$qr = $quantity * $rate'
color  shape    flag  k  index quantity rate   qr
yellow triangle true  1  11    43.6498  9.8870 431.5655726
red    square   true  2  15    79.2778  0.0130 1.0306114
red    circle   true  3  16    13.8103  2.9010 40.063680299999994
red    square   false 4  48    77.5542  7.4670 579.0972113999999
purple triangle false 5  51    81.2290  8.5910 697.8383389999999
red    square   false 6  64    77.1991  9.5310 735.7846221000001
purple triangle false 7  65    80.1405  5.8240 466.738272
yellow circle   true  8  73    63.9785  4.2370 271.0769045
yellow circle   true  9  87    63.5058  8.3350 529.3208430000001
purple square   false 10 91    72.3735  8.2430 596.5747605000001
```

https://github.com/johnkerl/miller

Работа с JSON

~$ sudo apt-get install jq

~$ cat file.txt | jq '.Title, .Year' # имеем дело с одним объектом

~$ cat file.txt | jq -c '.[].Title' # имеем дело со списком

~$ cat file.txt | jq --raw-input --slurp --arg home $HOME/project 'split("\n") | map(select(length > 0)) | . as $items | reduce range(0; $items|length) as $i ({}; . + { ($items[$i]): $home })'

http://hyperpolyglot.org/json

https://stedolan.github.io/jq/manual/

Реальный пайплайн

```
~$ cat file.txt | csvjson --stream | jq -c 'if .createdDate != "" then .createdDate =
(.standardRegCreatedDate | split(" ") | .[0:2] | join("T") + "Z" ) else .createdDate =
"9999-01-01T00:00:00Z" | to_entries | map(select(.key | contains("rawText") | not ) ) |
from_entries'
```

Параллелизация обработки

XArgs как магический инструмент

Подстановка результата выполнения одной команды в качестве аргумента другой команде

~$ find . -name "*.mp3" -print0 | xargs -0 ls

~$ find . -name "*.sh" -print0 | xargs -0 -I {} mv {} ~/back.scripts

~$ cat file.csv | csvcut -c "Field" | xargs # убираем переносы строк

~$ cat file.txt | xargs -I bash -c 'echo hdfs dfs -get $0 $1' | xargs -I {} -d '\n' -n1 -P8 -t bash -c "eval {}"

GNU Parallel — распределенка в консоли

~$ ls *.wav | parallel lame {} -o {}.mp3

~$ python makelist.py | parallel -j+2 'wget "{}" -O - | python parse.py'

~$ cat file.txt | awk '{ print "{\"index\": {} }","\n" $0 }' | parallel --pipe -N500 curl -s -XPOST localhost:9200/items/entry/_bulk --data-binary @- > /dev/null

man parallel_tutorial

Больше трюков!


```
~$ cat file.csv | parallel --colsep "\t" echo {2} {1} {3}

~$ cat file.csv | awk ‘!a[$1]++’  # sort -u не нужно

~$ cat file.csv | pv --line-mode -b > /dev/null  # ОЧЕНЬ КРУТО

~$ cat file.csv | peco | some_other_process  # выбор строк вручную

~$cat 76-0.txt | tr '[:upper:]' '[:lower:]' | grep -oE '\w+' | sort | uniq -c | sort -nr -k1,1 | head -n 50 | awk '{ print $2 "\t" $1 }' | gnuplot -p -e "set term png; set xtic rotate; plot '-' using (column(0)):2:xtic(1) smooth freq with boxes" > test.png
```

На почитать

https://github.com/jlevy/the-art-of-command-line

https://github.com/agarrharr/awesome-cli-apps

https://github.com/learn-anything/command-line-tools

https://xakep.ru/2016/05/17/console-magic/

https://xakep.ru/2016/06/07/console-magic-2/

Итоги. О чем поговорили:

1. Решение Ad-Hoc задач в консоли
2. Парсинг текстовых данных одной строкой
3. Запросы в API из терминала
4. Как работать с разными форматами
5. Как делать параллельные вычисления, не отходя от терминала

Спасибо
за внимание!