Boosting command line experience

Python meets AWK

Kirill Pavlov

Technical Recruiter, Terminal 1

14:40 – 15:10, November 05, 2017

README.md

- A lot of Python/AWK examples here.
- Source code and slides available online.
- At the end: build a stock trading system and check NYSE:CS.

Table of content

Table of content

Background

- Yandex, year 2010. Hadoop was not widly adopted.
- 10Gb of archived ads data daily: *time*, *ad_id*, *site_id*, *clicks*.
- Task: daily data aggregation (simple functions: group by, sum, join) and feature generation for further machine learning classification.
- Solution: released a set of command line scripts.

Example

This presentation uses UCI machine learning Higgs boson data: 11M objects, 28 attributes, 7.5Gb unarchived.

Questions:

1. What is the maximum value of *lepton_eta*?
2. What is the average value of *lepton_phi* by class 0 and 1?
3. Filter objects with *m_jj* > 0.75 (8.9M objects) and sort them by *m_wbb*.

Solutions:

1. In-memory Python with Pandas.
2. Databse SQL queries (PostgreSQL and Docker).
3. Command line with AWK.

Demo Time

Reality Check

It's not as agile as it seems. You work inside the company network.

1. You **don't have sudo** rights and your admin does not want to install anything for you. Like no database or user privileges, etc.
2. The **server does not have GitHub/Internet access** and the only deployment possible is Java JARs or C/C++/etc. So, no NodeJS/Python packages. And of course no R/Matlab/Excel.
3. Get better at command line tools ;)

Table of content

Basic concepts

1. AWK[1]— language for streaming columnar data processing. Standard in unix-like OS.
2. Actual AWK is outdated, use mawk (fast) or gawk (flexible).
3. Limited data structures: strings, **associative arrays (hash maps)** and regexps.
4. Built-in variables:
   - $1, $2, . . . ($0 is entire record)
   - NR - number of processe lines (records)
   - NF - number of columns (fields)
5. Use vars without declaration. Default values are 0s. One liners. **Hipster friendly**.

[1] Tutorial by Bruce Barnett. Careful, he writes his blog in txt

AWK Example 1

1.1 How many speakers does PyCon have (awk + sort)?

```
cat pycon-schedule.csv \
  | awk -F, '{if($4) print $4}' \
  | tail -n+2 | sort \
  | uniq | wc -l
```

1.2 How many speakers does PyCon have (tawk + tsort)?

```
cat pycon-schedule.tsv \
  | tawk -o 'speaker' -f 'speaker' \
  | tsrt -N \
  | uniq | wc -l
```

AWK Example 2

2. What is the most popular word in PyCon topics?

```
cut -d, -f5 pycon-schedule.csv \
  | tail -n+2 \
  | awk '{for(i=1; i<=NF; i++) w[\$i]++}END{for(i in w) print w[i], i}' \
  | sort -r | head
```

AWK Example 2

2. What is the most popular word in PyCon topics?

```
cut -d, -f5 pycon-schedule.csv \
  | tail -n+2 \
  | awk '{for(i=1; i<=NF; i++) w[\$i]++}END{for(i in w) print w[i], i}' \
  | sort -r | head
```

Python! (who could have guessed?)

AWK Example 3

3. Filter unique talks (one talk at a time)

```
cat pycon-schedule.tsv \
  | tawk -a -f 'speaker' \
    -o 'key=timeslot if day == 1 else gensub("-", "+", "g", timeslot)' \
  | tgrp -k key \
    -g 'day=FIRST(day)' -g 'timeslot=FIRST(timeslot)' \
    -g 'venue=FIRST(venue)' -g 'speaker=FIRST(speaker)' \
    -g 'title=FIRST(title)' -g 'sessions=COUNT()' \
  | tawk -o 'day;timeslot;venue;speaker;title' -f 'sessions==1'
```

Table of content

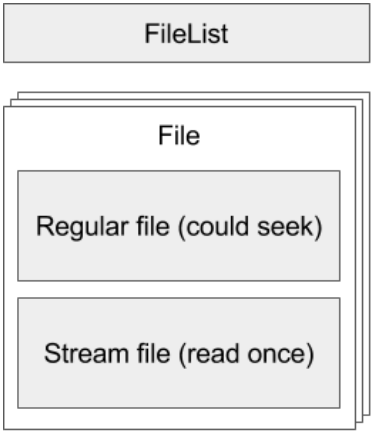
Basic concepts

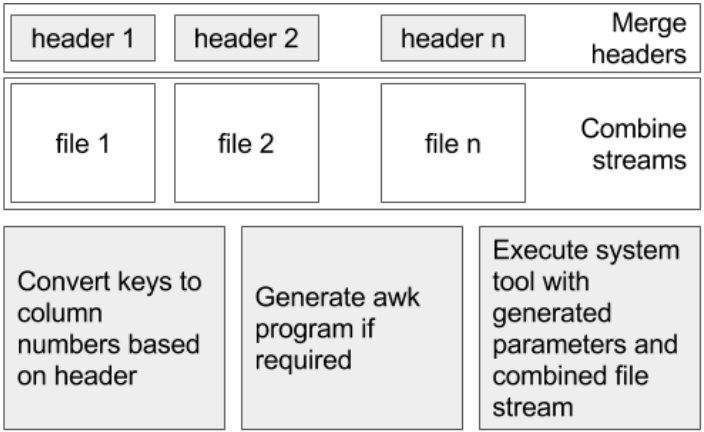
1. Special files format: tsv + header (meta information). Easy to convert and autogenerate headers.

```
# Date       Open    High    Low    Close   Volume
2014-02-21   84.35   84.45   83.9   83.45   17275.0
```

2. Python script manages file descriptors headers, convert column names to column numbers and executes command line command, e.g. cat/tail/sort.
3. Heavy lifting goes to awk: tawk (map) and tgrp (map-reduce).
4. Based on command line expressions, it generates awk command and executes it with incoming stream.
5. Visual sugar: tpretty and tplot.

Under the hood: FileList and stream management

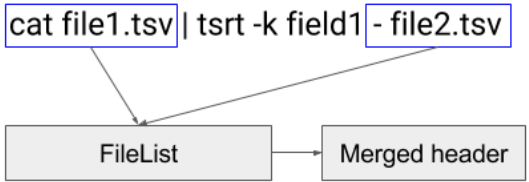


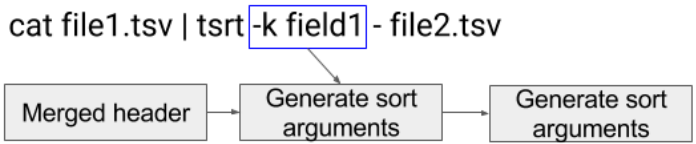


Under the hood: Argument parsing

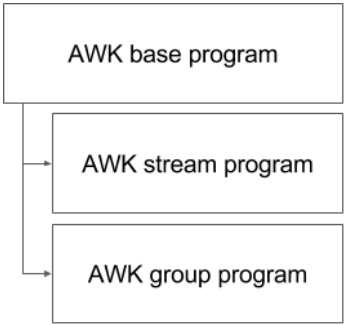
Consider bash command with two imput files and sort by one key:

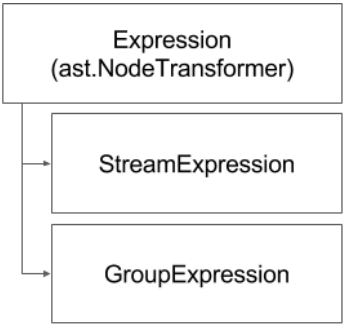
```
cat file1.tsv | tsrt -k field1 - file2.tsv
```





Under the hood: AWK program expressions





Assign, Name, Call,
Expr, If, Compare

SUM, SUM2, AVG,
EMA, MIN, MAX

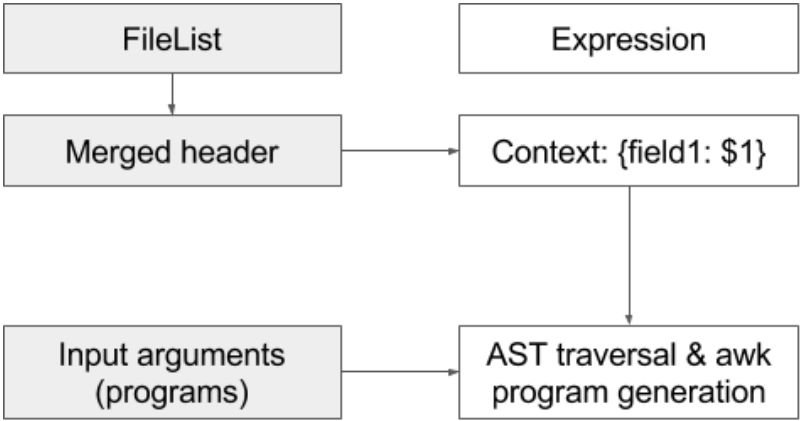
FIRST, LAST, MIN,
MAX, SUM, COUNT

List of Expressions:

1. BEGIN {}
2. {}
3. END {}

1. Generic expression converts Python code to awk.
2. Stream/Group extend python functions for awk.

Under the hood: Expression context



Homework

Given stream of data: 1, 2, 3, ... Write a program to calculate:

1. Streaming average S, such as S1=1, S2=(1+2)/2 S3=(1+2+3)/3
2. Running average A with window 3, such as A1=1, A2=S2, A4=(4+5+6)/3
3. Running maximum M with window 3, such as M1=1, M4=max(4, 5, 6)=6

Features

1. Streaming expressions: parametrized running/total sum/average/maximum[2].
2. Aggregators: first, last, min, max, count.
3. Modules: deque.
4. Build to self-contained 2k LOC portable python (2.7, 3.3+) scripts.
5. All together: zero-configuration extensible sql in command line. It is readable and faster than a generic python/cython code (even after shedskin) and perl.

[2] moving maximum in linear time with deque implemented on top of awk associative arrays.

Solutions comparison

Dell xps 15, 16Gb RAM, 8 CPUs:

	Python	**PostgreSQL**	**gawk**	**mawk**	**tabtools**
Read time	104.4	180.3	0	0	0
Q1: "max" time	0	15.2	22.8	12.2	12.8
Q2: "group + avg" time	0	5.8	30.5	12.6	26.6[3]
Q3: "filter + sort" time	21.3	33.6	174.2	36.3	33.5
Total, sec.	125.7	243.9	227.5	61.1	72.9

[3] Uses $\Omega(n \log(n))$ complexity instead of $\Omega(n)$. Could be improved.

Table of content

Data description

Credit Suisse (NYSE:CS) daily stock data from Yahoo Finance: 'CS.csv' + 'cs.tsv'.

```
cat cs.tsv | tgrp \
    -k "Week=strftime(\"%U\", DateEpoch(Date))" \
    -g "Date=FIRST(Date)" \
    -g "Open=FIRST(Open)" \
    -g "High=MAX(High)" \
    -g "Low=MIN(Low)" \
    -g "Close=LAST(Close)" \
    -g "Volume=SUM(Volume)" \
  | ttail \
  | tsrt -k Date:desc \
  | tpretty
```

Demo Time

Demo: metrics

1. Moving Average for windown size 200 and 50.
2. Exponential moving average for window size 26 and 13.
3. MACD(26, 12, 9) histogram.
4. Moving maximum and minimum for window size 14.
5. Fast and Slow Stochastics.

Demo: plot (expected and actual)

Thank you!

Kirill Pavlov <k@p99.io>, Recruiter, Terminal 1.
GitHub: @pavlov99 | Presentation: 2017-11-05-pycon | tabtools
#jobs https://t1.gl/k