Big Data Analytics (SOEN 498/691)

Laboratory sessions

Tristan Glatard
Department of Computer Science and Software Engineering
Concordia University, Montreal
tristan.glatard@concordia.ca

March 23, 2017

Contents

Part I

Log analysis

1 Introduction

In this assignment you will analyze log files using Apache Spark. We will focus on Linux syslog files and more precisely on the `messages` file located in `/var/log` on most Linux distributions. Such analyses have various applications in system administration, e.g., for security diagnostics.

Lines in `/var/log/messages` are formatted as follows:

```
<timestamp> <host_name> <source>:  <message>
```

where `<timestamp>` is a timestamp, `<host_name>` is the host (computer) name, `<source>` is the program that generated the line and `<message>` is the log message. Here is an example:

```
Feb 28 03:30:01 iliad systemd:  Started Session 4415 of user achille.
```

You have to write a Spark program to answer the **9 questions** detailed below (**Q1**-**Q9**). Log files collected on two hosts called `iliad` and `odyssey` are available on Moodle to test your program. During marking, your program will be run on similar but different files to check its correctness. Section 6 summarizes submission instructions and the marking scheme that will be used to evaluate your submission. You are strongly encouraged to review these instructions before starting the assignment.

2 Filtering and Counting

The following queries must be run separately on the log files of each host, i.e., *one answer must be returned for each host.*

Q1 - For each host, print the total number of available log lines.

Expected output on `iliad` and `odyssey`:

```
$ ./log_analyzer -q 1 iliad odyssey
* Q1: line counts
  + iliad: 63854
  + odyssey: 65405
```

Q2 - For each host, print the number of sessions that were started for user `achille`. Session starts are logged in messages containing the following string: `Starting Session <id> of user <user>`, where `<id>` is the session id and `<user>` is the user name.

Expected output on `iliad` and `odyssey`:

```
$ ./log_analyzer -q 2 iliad odyssey
* Q2: sessions of user ’achille’
  + iliad: 5173
  + odyssey: 5228
```

Q3 - For each host, list the unique user names who started a session.

Expected output on iliad and odyssey:

```
$ ./log_analyzer -q 3 iliad odyssey
* Q3: unique user names
  + iliad: ['gaia', 'pollux', 'achille', 'helene', 'hector']
  + odyssey: ['achille', 'hector', 'ares']
```

Q4 - For each host, list the number of sessions started per user.

Expected output on iliad and odyssey:

```
$ ./log_analyzer -q 4 iliad odyssey
* Q4: sessions per user
  + iliad: [('gaia', 2), ('pollux', 38), ('achille', 5173), ('helene', 248), ('hector', 9)]
  + odyssey: [('achille', 5228), ('hector', 2), ('ares', 40)]
```

3 Errors

The following queries must be run separately on the logs of each host, i.e., *one answer must be returned for each host.*

Q5 - For each host, count the error messages, i.e., the lines that contain string "error" (case insensitive match).

Expected output on iliad and odyssey:

```
$ ./log_analyzer -q 5 iliad odyssey
* Q5: number of errors
  + iliad: 2723
  + odyssey: 25805
```

Q6 - For each host, print the 5 most frequent error messages and their counts.

Expected output on iliad and odyssey:

```
$ ./log_analyzer -q 6 iliad odyssey
* Q6: 5 most frequent error messages
  + iliad:
    - (889, 'journal: ethtool ioctl error: No such device ')
    - (24, 'gnome-session: ** (evince:31187): WARNING **: Error setting file metadata: No such file or directory ')
    - (9, 'gnome-session: https://yum.dockerproject.org/repo/main/centos/7/repodata/3849c07e5505140b8134d3cf1bef35c
    - (9, "gnome-session:␣GDBus.Error:org.gtk.GDBus.UnmappedGError.Quark._imsettings_2derror_2dquark.Code5:␣Current
    - (8, 'firefox.desktop: Crash Annotation GraphicsCriticalError: |[0][GFX1-]: GLContext is disabled due to a pre
  + odyssey:
    - (9229, 'gnome-session: (tracker-miner-fs:30474): Tracker-CRITICAL **: Could not execute sparql: column nie:ur
    - (4519, 'gnome-session: (tracker-miner-fs:30474): Tracker-CRITICAL **: (Sparql buffer) Error in task 0 of the
    - (2776, 'gnome-session: (tracker-miner-fs:30474): Tracker-CRITICAL **: (Sparql buffer) Error in task 2 of the
    - (2401, 'gnome-session: (tracker-miner-fs:1259): Tracker-CRITICAL **: Could not execute sparql: column nie:url
    - (1697, 'gnome-session: (tracker-miner-fs:1259): Tracker-CRITICAL **: (Sparql buffer) Error in task 0 of the a
```

4 Combining logs

The following queries are run on a combination of both hosts, i.e., *a single answer combining information coming from both hosts must be returned for each question.*

Q7 - List the user names who started a session on both hosts.

Expected output on `iliad` and `odyssey`:

```
$ ./log_analyzer -q 7 iliad odyssey
* Q7: users who started a session on both hosts, i.e., on exactly 2 hosts.
  + : ['achille', 'hector']
```

Q8 - List the user names who started a session on exactly one host and list this host.

Expected output on `iliad` and `odyssey`:

```
$ ./log_analyzer -q 8 iliad odyssey
* Q8: users who started a session on exactly one host, with host name.
  + : [('gaia', 'iliad'), ('pollux', 'iliad'), ('helene', 'iliad'), ('ares', 'odyssey')]
```

5 Anonymization

Q9 - Anonymize the logs, i.e., replace user names with strings formatted as `user-<i>` where `i` is the index of the user name in the array of original user names sorted alphabetically. For instance, user names `foo` and `bar` must be replaced by `user-1` and `user-0` (respectively), assuming that `foo` and `bar` are the only user names in the log. In addition to writing the anonymized files, your program must print the mapping used for the anonymization and the location where the anonymized files were written.

Expected output on `iliad` and `odyssey`:

```
$ ./log_analyzer -q 9 iliad odyssey
  + iliad:
  . User name mapping: [('achille', 'user-0'), ('gaia', 'user-1'), ('hector', 'user-2'),\
    ('helene', 'user-3'), ('pollux', 'user-4')]
. Anonymized files: iliad-anonymized-10
  + odyssey:
. User name mapping: [('achille', 'user-0'), ('ares', 'user-1'), ('hector', 'user-2')]
. Anonymized files: odyssey-anonymized-10
```

The corresponding anonymized logs are available on Moodle.

6 Submission and Marking

Your assignment must be submitted as a `tgz` or `zip` archive on Moodle. The archive name must contain your name(s) and student id(s). If you are submitting in a team of two, each team member must submit a copy of the assignment. The submitted archive has to include the following files:

- A command-line program that prints the output of the above questions to the console, using the following syntax:

  ```
  $ ./log_analyzer -q <i> <dir1> <dir2>
  ```

  Where `<dir1>` and `<dir2>` are two directories containing log files collected on two different hosts, and `<i>` is the question number. For instance, your program will be executed as follows to run question 7 on the test logs:

  ```
   $ log_analyzer -q 7 iliad odyssey
  ```

- The source code of your program. You may use any programming language supported by Spark, i.e., Java, Scala or Python.

- A README file explaining how to compile (if relevant) your program, and any other useful information to run it, e.g., required versions, required environment variables, etc.

- A Git commit history (`.git` directory) or the URL of a Git repository (e.g., Github) that you used to develop your program. Your commit history will be checked and frequent commits showing incremental development will be favored over monolithic ones.

Your program will be run on the testing data (`iliad` and `odyssey`) and on undisclosed `/var/log/messages` files collected on Linux hosts. The following marking scheme will be used:

- 5 points: program compiles (if relevant) and runs with the imposed syntax.

- For each question (9 questions in total):
  - 2 points: program runs to completion.
    - ∗ Program runs to completion out of the box: 2 points.
    - ∗ Program runs to completion after minor fixes: 1 point.
    - ∗ Program runs to completion after major fixes or cannot be fixed: 0 point.
  - 2 points: program gives correct answer on testing data (`iliad` and `odyssey`).
    - ∗ Program gives totally correct answer: 2 points.
    - ∗ Program gives partially correct answer (e.g. incomplete lists): 1 point.
    - ∗ Program gives incorrect answer: 0 point.
  - 4 points: program gives correct answer on undisclosed evaluation data.
    - ∗ Program gives totally correct answer: 4 points.
    - ∗ Program gives partially correct answer: 2 points.
    - ∗ Program gives incorrect answer: 0 point.
  - 2 points: program uses mainly Spark operations (transformation and actions).
    - ∗ Program uses mainly Spark operations: 2 points.
    - ∗ Program uses some Spark operations: 1 point.
    - ∗ Program does not use Spark operations: 0 point.

- 5 points: Git commit history is available and shows frequent commits.

- Commit history is available and demonstrates frequent commits showing incremental development steps: 5 points.
- Commit history is available but shows only a few large commits: 2 points.
- Commit history is not available: 0 point.

Total: 100 points (5+9×10+5).