Master Thesis Project Description

Topic: Apps Frequently Installed Together

42matters AG is a rapidly growing startup, leading the development of next generation mobile user modeling technology. Our solutions are used by big brand companies within the mobile advertising market to serve mobile users intelligently targeted content. We are an international team, with an innovative and fast-paced company culture.

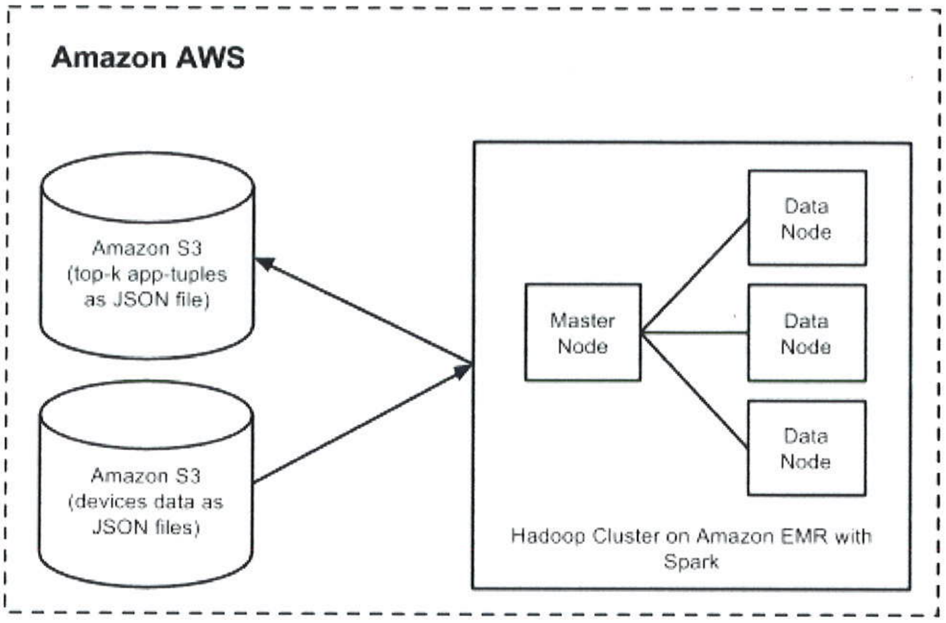
Project Description

As of today on Google Play there exist about 1 million apps. Few of them have many installations and a vast amount of them only few. The goal of the project is to implement an algorithm for discovering which apps are frequently installed together on mobile devices. A challenge in implementing the algorithm is to handle the large amount of app-tuples which in the case of app-pairs (i.e. app-2-tuples) in the worst case can be 1 million squared. The final result will be a list of the top-k most frequently installed together app-tuples for each app.

The algorithm has to be implemented using Hadoop/Spark/Python on Amazon Elastic MapReduce (Amazon EMR). Details about the project are presented in the following sections.

System Architecture

In this section the overall architecture of the system is presented. In the figure below an overview of the systems involved in the project is provided. The system will reside completely on Amazon Web Services (Amazon AWS). The input data will be stored on Amazon S3 in JSON file format. The algorithms will run on Amazon Elastic MapReduce (Amazon EMR) and will be implemented as Spark programs based on Python.


Amazon AWS
Amazon S3 (top-k app-tuples as JSON file)
Amazon S3 (devices data as JSON files)
Master Node
Data Node
Data Node
Data Node
Hadoop Cluster on Amazon EMR with Spark

Input Data Structure

The input data used in the project is data about mobile devices and about the apps installed on them. The data is stored as JSON files on Amazon S3. In the following, details about the major fields of the data sources are described.

- Devices

  This data source contains data about mobile devices. Each line of the JSON file stores one device. The device is encoded as a JSON document containing among others the following fields:
  - udid: The unique identifier of a device
  - country: The country of the device
  - timestamp: Timestamp of the last update of the document
  - apps: List of apps installed on the device (apps are identified by their package name)
    - pn: The package name, i.e. unique identifier, of the installed app.

  The following example represents a device (note that in the JSON file this device document will be stored in one line):

```
{
        udid : "++/OarsCrkiQx5EyY/XTVxOwc4m1H2Re3m+CdiW+YeU=",
        country : "CH",
        timestamp : ISODate("2014-09-24T10:18:56.531Z"),
        apps : [
               {
                      fit : ISODate("2013-10-12T03:47:39Z"),
                      lut : ISODate("2014-6-22T12:15:19Z"),
                      pn :"playboard.android"
               },
               {
                      fit : ISODate("2010-05-18T08:43:32Z"),
                      lut : ISODate("2014-09-24T10:11:46Z"),
                      pn :"com.facebook.katana"
               },
               ...
        ],
        ...
}
```
This data can grow to many millions of devices.

Top-k Most Frequently Installed App-Tuples Algorithm

The goal of the top-k most frequently installed app-tuples algorithm is to find which apps are often installed together on mobile devices. For instance it could be that mobile devices, which have the app Facebook (in the following referred to as "subject app") installed, have frequently also WhatsApp and Twitter installed. So for instance if considering app pairs (i.e. 2-tuples) for Facebook we would have the pairs (Facebook, WhatsApp) and (Facebook, Twitter) among the top-k pairs. The same could be done for triples, e.g. (WhatsApp, Facebook, Twitter), on n-tuples, e.g. (WhatsApp, Facebook Twitter, ...).

In the following we illustrate the main steps involved in computing the top-3 app pairs (2-tuples) for Facebook app:

- Count number of devices having a Facebook installed together with other apps. We refer to that number as device app tuple frequency (DATF). The DATF has to be computed for all pairs of apps appearing on any of the devices.

Package name #1	Package name #2	Count (DATF)
com.facebook.katana	com.whatsapp	126,000
com.facebook.katana	com.twitter.android	143,000
com.facebook.katana	com.skype.raider	94,500
com.facebook.katana	com.google.android.talk	102,600

...	...	...

- Rank the apps based on their DATF and keep only the top-3 pairs. In the following the top-3 paris for the Facebook app are Twitter (com.twitter.android), WhatsApp (com.whatsapp), and Google Hangouts (com.google.android.talk). The app Skype (com.skype.raider) is not among the top-3 pairs anymore and thus discarded.

Package name #1	Package name #2	Score
com.facebook.katana	com.twitter.android	143,000
com.facebook.katana	com.whatsapp	126,000
com.facebook.katana	com.google.android.talk	102,600
~~com.facebook.katana~~	~~com.skype.raider~~	~~94,500~~

Project Tasks

1. Implementation of the top-k app pairs in Spark for Amazon AWS.
    - Brute-force algorithm implementation which for each app computes the top-k apps most frequently installed together with that app.
    - Possibility to configure k.
    - Possibility to filter n-tuples based on a minimum ratio between the DATF and the number of installation of the "subject app".
    - The implementation has to be done with Spark in Python and it must be possible to run the code on Amazon EMR.
    - Documentation of the program and of how to run the program on Amazon EMR.

2. Investigation of optimisations of the brute force top-k algorithm.
    - Investigations of possible optimisations of the top-k brute force algorithm, for instance one possible optimisation could be to prune pairs for which the DAF of the pairing app is smaller than the the DAPF of the current top-k pairs. The student has to autonomously investigate this and/or other possible optimisations and to adapt the algorithm accordingly.
    - Implementation of a new version/versions of the algorithm with Spark in Python. It must be possible to run the code on Amazon EMR.
    - Documentation of the program and of how to run the program on Amazon EMR.

3. Extension of the algorithms to triples or n-tuples.
    - The algorithms have to be adapted in order to allow to compute top-k triples or top-k n-tuples (where n can be an integer value >= 2). Note that app pairs are 2-tuples, app triples 3-tuples and so on.
    - Implementation of the new algorithm for triples or n-tuples with Spark in Python. It must be possible to run the code on Amazon EMR.

4. Experiments planning
    - An experimental evaluation of the top-k app paris / triples / n-tuples algorithms has to be done highlighting performance, scalability and cost aspects.
    - In order to run the experiments the student has to generate synthetic data. This could be done by implementing a program which generates devices and apps which reflect the real world distributions of app installation.
    - Experiments with up to 100 million devices and about 1 million apps should be planned.

Notes:

- All parts of the project (code, algorithms, etc.) have to be documented so that it is possible to understand them and to reproduce

Challenges

- Understanding and using the technology stack
- Algorithms definition for top-k app n-tuples
- Experimental evaluation

What 42matters Provides

- Provide budget for Amazon AWS
- Provide AWS account
- Introduction and materials to kickstart
    - How to use AWS
    - “Hello world” program running Spark on AWS
- Small amount of real data, and data distribution to generate synthetic data
- Requirements on input and output data formats
- Biweekly meeting (about 2 hours)
- Feedbacks on the thesis

Supervisor at 42matters

Christian Ammendola
christian@42matters.com