

Crunching and visualizing Big Data on a Computer Cluster

Joana Simões

March 30, 2015

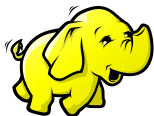


Table of Contents

- 1 Introduction
- 2 Importing a Spatial-Temporal Series
- 3 Recovering the Spatial Attributes
- 4 Putting it All Together
- 5 Piping the Results into the Outside World

Motivation

- Problem: the increasing volume of information, by explosion of traditional sources + new sources
- Target: fast query responses, which require a scalable architecture
- Possible solution: support clusters on a cost-effective architecture, such as commodity clusters or cloud environments



Importing a Spatial-Temporal Series Recovering the Spatial Attributes Putting it All Together Piping the Results into the Outside World

Cloud Services

Compute

Amazon Elastic Compute Cloud (Amazon EC2)

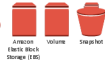


Storage

Amazon Simple Storage Service (Amazon S3)



Amazon Elastic Block Storage (Amazon EBS)



AWS Import/Export



AWS Storage Gateway Service



AWS Glacier



Database

Amazon DynamoDB



Amazon Relational Database Service (Amazon RDS)



Amazon ElastiCache



Networking

Amazon Route 53



Amazon Elastic Load Balancing



AWS Direct Connect



Amazon Virtual Private Cloud (VPC)



Content Delivery

Amazon Cloudfront



Elastic Network Instance



Application Services

Amazon Simple Queue Service (SQS)



Amazon Cloudsearch



Amazon Simple Email Service (SES)



Amazon Simple Workflow (SWF)



Amazon Simple Notification Service (SNS)



Deployment and Management

Amazon Elastic Beanstalk



AWS Identity and Access Management (IAM)



AWS CloudFormation



Monitoring

Amazon CloudWatch



Non-Service Specific



Groups

A thought...

First the use case, then the tools.

Use Case

- Study spatial and temporal patterns of road traffic accidents.
- Relate target variable (accident) with context variables (e.g.: weather, proximity to SPI).

Use Case

- Study spatial and temporal patterns of road traffic accidents.
- Relate target variable (accident) with context variables (e.g.: weather, proximity to SPI).
- In most (Big) Data Analysis 80% of the effort is devoted to the Extract-Transform-Load (ETL) process
- ETL: process responsible for pulling data out of the source systems and placing it into a data warehouse

Use Case

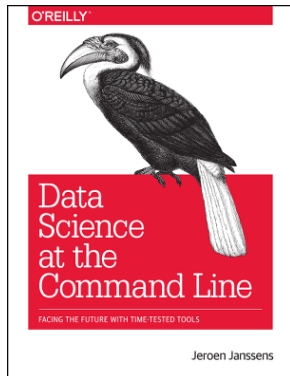
- Study spatial and temporal patterns of road traffic accidents.
- Relate target variable (accident) with context variables (e.g.: weather, proximity to SPI).
- In most (Big) Data Analysis 80% of the effort is devoted to the Extract-Transform-Load (ETL) process
- ETL: process responsible for pulling data out of the source systems and placing it into a data warehouse
 - **Extract** data from different source systems and convert it into one consolidated data warehouse format which is ready for transformation processing
 - **Transform**: cleaning, filtering, splitting a column, joining data, apply validation, apply rules, etc
 - **Load**: into the data warehouse, repository or reporting applications

Another thought...

There are no free lunches.

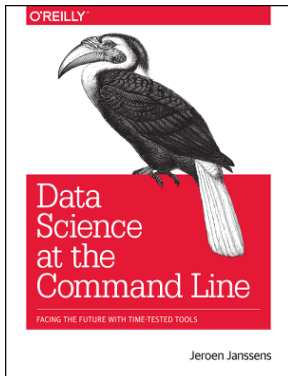
Scalable ML Platforms

- ML platforms may provide high-level data import tools:



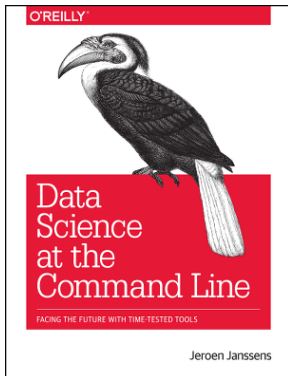
Scalable ML Platforms

- ML platforms may provide high-level data import tools:
 - They generally trade ease of use for flexibility



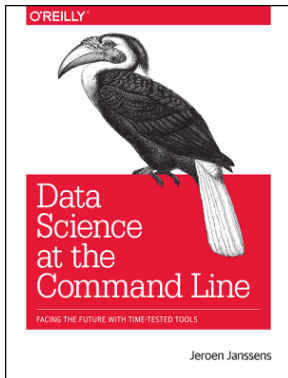
Scalable ML Platforms

- ML platforms may provide high-level data import tools:
 - They generally trade ease of use for flexibility
 - They may have a cost (\$\$\$)

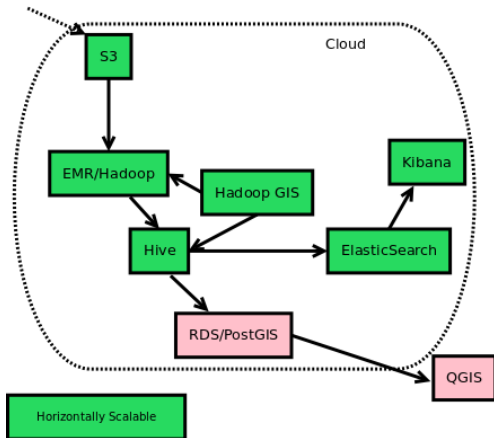


Scalable ML Platforms

- ML platforms may provide high-level data import tools:
 - They generally trade ease of use for flexibility
 - They may have a cost (\$\$\$)
- To ensure maximum flexibility, we should be able to link together many tools, often using the command line.



Stack



Importing a Spatial-Temporal Series
Recovering the Spatial Attributes
Putting it All Together
Piping the Results into the Outside World

Hadoop

Importing a Spatial-Temporal Series
Recovering the Spatial Attributes
Putting it All Together
Piping the Results into the Outside World

Hive

Importing a Spatial-Temporal Series
Recovering the Spatial Attributes
Putting it All Together
Piping the Results into the Outside World

Hadoop GIS

Importing a Spatial-Temporal Series
Recovering the Spatial Attributes
Putting it All Together
Piping the Results into the Outside World

Pig

Importing a Spatial-Temporal Series
Recovering the Spatial Attributes
Putting it All Together
Piping the Results into the Outside World

PostgreSQL + PostGIS

Importing a Spatial-Temporal Series
Recovering the Spatial Attributes
Putting it All Together
Piping the Results into the Outside World

E(L)K

From S3 to HDFS

- Micro-task: Understand the dataset structure
 - A sample dataset is stored on an S3 bucket:
`s3n://workshop-bdsd/accidents/`

From S3 to HDFS

- Micro-task: Understand the dataset structure
 - A sample dataset is stored on an S3 bucket:
`s3n://workshop-bdsd/accidents/`
 - `https://s3-eu-west-1.amazonaws.com/workshop-bdsd/accidents/accidents_sample.csv`

From S3 to HDFS

- Micro-task: Understand the dataset structure
 - A sample dataset is stored on an S3 bucket:
`s3n://workshop-bdsd/accidents/`
 - `https://s3-eu-west-1.amazonaws.com/workshop-bdsd/accidents/accidents_sample.csv`
 - Download and view dataset

From S3 to HDFS (cont.)

- Micro-task: Create a table linking to the data
 - Enter hive and create an external table linking to the S3 bucket

From S3 to HDFS (cont.)

- Micro-task: Create a table linking to the data
 - Enter hive and create an external table linking to the S3 bucket
 - Use the CSV serde to parse the table structure

From S3 to HDFS (cont.)

- Micro-task: Create a table linking to the data
 - Enter hive and create an external table linking to the S3 bucket
 - Use the CSV serde to parse the table structure
 - separator char
 - quote char
 - headers

From S3 to HDFS (cont.)

- Micro-task: Create a table linking to the data
 - Enter hive and create an external table linking to the S3 bucket
 - Use the CSV serde to parse the table structure
 - separator char
 - quote char
 - headers
 - View imported data

From S3 to HDFS (cont.)

- Micro-task: Type Mapping

From S3 to HDFS (cont.)

- Micro-task: Type Mapping
 - Create an empty table with correct types

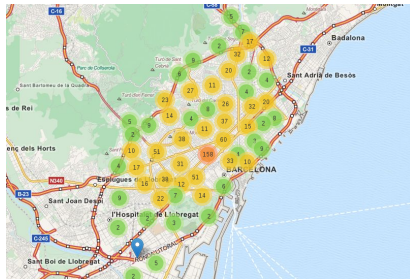
From S3 to HDFS (cont.)

- Micro-task: Type Mapping
 - Create an empty table with correct types
 - Insert data from `accidents_import`

From S3 to HDFS (cont.)

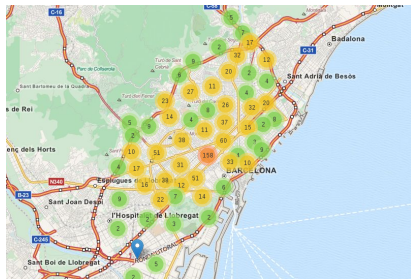
- Micro-task: Type Mapping
 - Create an empty table with correct types
 - Insert data from accidents_import
 - View table

What is so "Special" about Spatial



What is so "Special" about Spatial

- Location attributes allow us to detect spatial patterns
- Location also works as a "key", allowing us to connect with other datasets



Analysis of the Spatial Attributes

- Spatial Attributes are encoded as coordinates in "d_coord_geo_impacte"

Analysis of the Spatial Attributes

- Spatial Attributes are encoded as coordinates in "d_coord_geo_impacte"
- Problems with this field:

Analysis of the Spatial Attributes

- Spatial Attributes are encoded as coordinates in "d_coord_geo_impacte"
- Problems with this field:
 - Unstructured: needs parsing
 - Inconsistent format (order of coordinates, separator)
 - Invalid values (e.g.: 77, names)
 - No metadata

Analysis of the Spatial Attributes

- Spatial Attributes are encoded as coordinates in "d_coord_geo_impacte"
- Problems with this field:
 - Unstructured: needs parsing
 - Inconsistent format (order of coordinates, separator)
 - Invalid values (e.g.: 77, names)
 - No metadata
 - Mixed CRS:

Analysis of the Spatial Attributes

- Spatial Attributes are encoded as coordinates in "d_coord_geo_impacte"
- Problems with this field:
 - Unstructured: needs parsing
 - Inconsistent format (order of coordinates, separator)
 - Invalid values (e.g.: 77, names)
 - No metadata
 - Mixed CRS:
 - WGS84 (EPSG:4326)
 - European Grid (EPSG:5554)
 - European Grid encoded by the police using an *ad-hoc* format

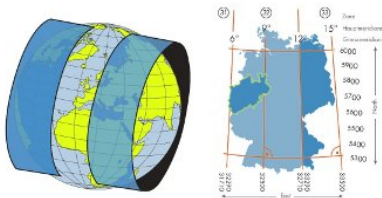
Analysis of the Spatial Attributes

- Spatial Attributes are encoded as coordinates in "d_coord_geo_impacte"
- Problems with this field:
 - Unstructured: needs parsing
 - Inconsistent format (order of coordinates, separator)
 - Invalid values (e.g.: 77, names)
 - No metadata
 - Mixed CRS:
 - WGS84 (EPSG:4326)
 - European Grid (EPSG:5554)
 - European Grid encoded by the police using an *ad-hoc* format
 - $lon = y/1000 + 400000$
 - $lat = y/1000 + 4500000$

CRS

World Geodetic System (WGS84, EPSG:4326): standard for use in cartography, geodesy, and navigation; reference CRS for GPS.

European Terrestrial Reference System 1989 (ETRS89, EPSG:5554): proposed, multipurpose Pan-European mapping standard; based on the ETRS89 Lambert Azimuthal Equal-Area projection coordinate reference system



Objective

- Separate lat, long fields and map them to correct types
- Remove invalid values
- Convert all coordinates into a single CRS (WGS84)

Exporting Data to Pig

- Pig uses filters to subset the data
- To merge back the subsetting data, we can use joins by a common field
- Micro-task: Export the data

Exporting Data to Pig

- Pig uses filters to subset the data
- To merge back the subsetting data, we can use joins by a common field
- Micro-task: Export the data
 - Create a copy of the accidents table, with an id field (joins).

Exporting Data to Pig

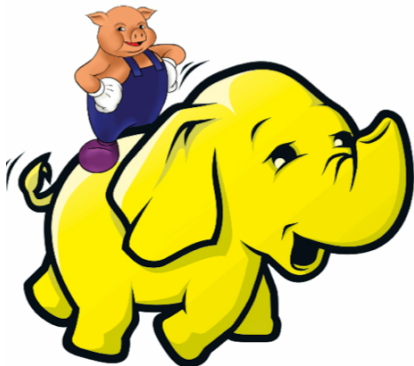
- Pig uses filters to subset the data
- To merge back the subsetting data, we can use joins by a common field
- Micro-task: Export the data
 - Create a copy of the accidents table, with an id field (joins).
 - Export this table into a tsv

Exporting Data to Pig

- Pig uses filters to subset the data
- To merge back the subsetting data, we can use joins by a common field
- Micro-task: Export the data
 - Create a copy of the accidents table, with an id field (joins).
 - Export this table into a tsv
 - Store it in HDFS (if needed)
 - View exported data

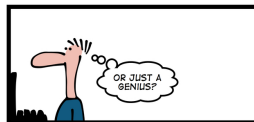
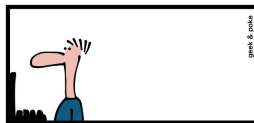
Presenting the Pig Script

- Subsets the coordinate list, using filters
- Detects each coordinate "type", using regular expressions
- In the case of grid encoded, it applies a formula to decode back into grid
- Stores the results into separate files, in HDFS



REGEX

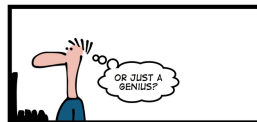
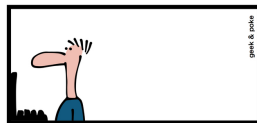
- Sequence of characters that forms a search pattern, mainly for use in pattern matching with strings, or string matching



YESTERDAYS REGEX

REGEX

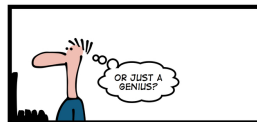
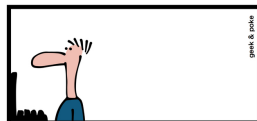
- Sequence of characters that forms a search pattern, mainly for use in pattern matching with strings, or string matching
- `REGEX_EXTRACT(D_COORD_Cz]',0)`



YESTERDAYS REGEX

REGEX

- Sequence of characters that forms a search pattern, mainly for use in pattern matching with strings, or string matching
- `REGEX_EXTRACT(D_COORD_Cz]',0)`
- `'[A-z]'`



YESTERDAYS REGEX

Running Pig

- Micro-task: run pig script

Running Pig

- Micro-task: run pig script
 - Download script from S3: https://s3-eu-west-1.amazonaws.com/workshop-bdsd/recover_geography.pig

Running Pig

- Micro-task: run pig script
 - Download script from S3: `https://s3-eu-west-1.amazonaws.com/workshop-bdsd/recover_geography.pig`
 - Edit script and **ammend paths**

Running Pig

- Micro-task: run pig script
 - Download script from S3: `https://s3-eu-west-1.amazonaws.com/workshop-bdsd/recover_geography.pig`
 - Edit script and **ammend paths**
 - Run script

Running Pig

- Micro-task: run pig script
 - Download script from S3: `https://s3-eu-west-1.amazonaws.com/workshop-bdsd/recover_geography.pig`
 - Edit script and **ammend paths**
 - Run script
 - Check output files

Importing Data Back into Hive

- Micro-task: Create tables linking to pig output

Importing Data Back into Hive

- Micro-task: Create tables linking to pig output
 - Create table with wgs84 data
 - Create table with grid data
 - Create table with police-decoded data

Exporting data into PostGIS

- As of Hadoop GIS 2.0, CRS transformation is **not supported**
- We need to rely on another tool: PostGIS on RDS
- Micro-task: Export grid data to PostGIS



Exporting data into PostGIS

- As of Hadoop GIS 2.0, CRS transformation is **not supported**
- We need to rely on another tool: PostGIS on RDS
- Micro-task: Export grid data to PostGIS
 - Merge grid (grid + police) tables in a single table



Exporting data into PostGIS

- As of Hadoop GIS 2.0, CRS transformation is **not supported**
- We need to rely on another tool: PostGIS on RDS
- Micro-task: Export grid data to PostGIS
 - Merge grid (grid + police) tables in a single table
 - Exported merged table into TSV



Importing Data into PostGIS

- Micro-task: Import grid data into PostGIS

Importing Data into PostGIS

- Micro-task: Import grid data into PostGIS
 - Install the PSQL client
 - Log into RDS:

Importing Data into PostGIS

- Micro-task: Import grid data into PostGIS
 - Install the PSQL client
 - Log into RDS:
 - host: `bdigitaldb.celqzuwfokoe.eu-west-1.rds.amazonaws.com`
 - user: `workshop`
 - password: `geohipster`
 - database: `workshop_bdigital`

Importing Data into PostGIS

- Micro-task: Import grid data into PostGIS
 - Install the PSQL client
 - Log into RDS:
 - host: `bdigitaldb.celqzuwfokoe.eu-west-1.rds.amazonaws.com`
 - user: `workshop`
 - password: `geohipster`
 - database: `workshop_bdigital`
 - Create table to accomodate data

Importing Data into PostGIS

- Micro-task: Import grid data into PostGIS
 - Install the PSQL client
 - Log into RDS:
 - host: `bdigitaldb.celqzuwfokoe.eu-west-1.rds.amazonaws.com`
 - user: `workshop`
 - password: `geohipster`
 - database: `workshop_bdigital`
 - Create table to accomodate data
 - Copy data into table

CRS Transformation

- Micro-task: Convert all features in European grid to WGS84

CRS Transformation

- Micro-task: Convert all features in European grid to WGS84
 - Create geometry fields to accomodate geometry in the two CRS (Grid, WGS84)

CRS Transformation

- Micro-task: Convert all features in European grid to WGS84
 - Create geometry fields to accomodate geometry in the two CRS (Grid, WGS84)
 - Add columns
 - Set SRID
 - Create geometry index

CRS Transformation

- Micro-task: Convert all features in European grid to WGS84
 - Create geometry fields to accomodate geometry in the two CRS (Grid, WGS84)
 - Add columns
 - Set SRID
 - Create geometry index
 - Instantiate grid geometry

CRS Transformation

- Micro-task: Convert all features in European grid to WGS84
 - Create geometry fields to accomodate geometry in the two CRS (Grid, WGS84)
 - Add columns
 - Set SRID
 - Create geometry index
 - Instantiate grid geometry
 - Transform grid geometry into another CRS

CRS Transformation

- Micro-task: Convert all features in European grid to WGS84
 - Create geometry fields to accomodate geometry in the two CRS (Grid, WGS84)
 - Add columns
 - Set SRID
 - Create geometry index
 - Instantiate grid geometry
 - Transform grid geometry into another CRS
 - Export grid geometry in GeoJSON

GeoJSON

GeoJSON: is an open standard format for encoding collections of simple geographical features along with their non-spatial attributes using JavaScript Object Notation.

The screenshot shows the GeoJSONLint website (geojsonlint.com). The interface includes a navigation bar with links for Point, LineString, Polygon, Feature, FeatureCollection, and GeometryCollection. Below the navigation bar, there is a text input area containing a GeoJSON FeatureCollection. The input text is as follows:

```
{
  "type": "Point",
  "id": "Sydney, Australia",
  "properties": {},
  "type": "Feature"
},
{
  "geometry": {
    "coordinates": [ 18.4172485,
      -33.9289049
    ],
    "type": "Point",
    "id": "Cape Town, South Africa",
    "properties": {},
    "type": "Feature"
  },
  "geometry": {
    "coordinates": [
      [151.2164539, -33.8548157],
      [18.4172485, -33.9289049]
    ],
    "type": "LineString",
    "id": null,
    "properties": {},
    "type": "Feature",
    "type": "FeatureCollection"
  }
}
```

Below the input area, there are two buttons: "Test GeoJSON" and "Clear". A checkbox labeled "Clear Current Features" is also present. To the right of the input area is a world map showing the locations of the features: Sydney, Australia (blue pin), Cape Town, South Africa (blue pin), and a line connecting them (blue line). The map is powered by Leaflet, with tiles courtesy of MapQuest and map data from OpenStreetMap contributors.

Importing Data back into Hive

- Micro-task: Import transformed data

Importing Data back into Hive

- Micro-task: Import transformed data
 - Enter Hive

Importing Data back into Hive

- Micro-task: Import transformed data
 - Enter Hive
 - Create table linking to the PostGIS export

Importing Data back into Hive

- Micro-task: Import transformed data
 - Enter Hive
 - Create table linking to the PostGIS export
 - Create new table and instantiate geometry from GeoJSON

Joining Data

- Micro-task: Join imported coordinates with WGS84 coordinates and the rest of the dataset

Joining Data

- Micro-task: Join imported coordinates with WGS84 coordinates and the rest of the dataset
 - Join imported records with original table with all fields

Joining Data

- Micro-task: Join imported coordinates with WGS84 coordinates and the rest of the dataset
 - Join imported records with original table with all fields
 - Merge imported records with WGS84 records, for a single table with unified geometry

Understanding Indexes

Generating Indexes

References

- <http://tweettracker.fulton.asu.edu/tda/TwitterDataAnalytics.pdf>
- <http://www2.qgis.org>
- <http://plugins.qgis.org/plugins/>
- <http://geokoder.com/mongodb-plugin-for-quantum-gis>
- <http://www.gislounge.com/heat-maps-in-gis/>
- <https://alastaira.wordpress.com/2011/02/23/heat-mapping-crime-data-with-bing-maps-and-html5-canvas/>
- http://docs.qgis.org/2.0/en/docs/user_manual/plugins/plugins_heatmap.html
- http://en.wikipedia.org/wiki/Kernel_%28statistics%29#Kernel_functions_in_common_use
- http://en.wikipedia.org/wiki/Cluster_analysis
- https://plugins.qgis.org/plugins/clusterpy_qgis_plugin/
- <http://www.rise-group.org/section/Software/clusterPy/>
- <http://threejs.org/>
- <http://anitagraser.com/2014/03/15/3d-viz-with-qgis-three-js/>

Thank you for Listening!

