Querying ~~1.6~~ 1.8 billion reddit comments with python

Daniel Rodriguez / PyData NYC / Nov 11, 2015

github.com/danielfrg/pydata-nyc-2015 (https://github.com/danielfrg/pydata-nyc-2015)

About me

Daniel Rodriguez

Data Scientist and Software Developer at Continuum Analytics

Anaconda Cluster

Bogotá, Colombia [1] now in Austin, TX

- twitter.com/danielfrg (twitter.com/danielfrg)
- github.com/danielfrg (github.com/danielfrg)
- danielfrg.com (danielfrg.com)

---

[1] Yes, it’s spelled like that, with two ‘o’s. Columbia with a ‘u’ is a university in New York City. It’s not that hard: ColOmbia is a cOuntry, ColUmbia is a University.

What is this talk about?

Querying 1.8 billion reddit comments with python

And how you can do it

- Data science "friendly" clusters
- ETL plus a little bit on data formats
  - Parquet
- Querying data with some new python libraries that target remote engines
  - Impala

Assumes some basic knowledge of big data tools like HDFS, Map Reduce, Spark or similar

More info at: http://blaze.pydata.org/blog/2015/09/16/reddit-impala/
(http://blaze.pydata.org/blog/2015/09/16/reddit-impala/)

Data

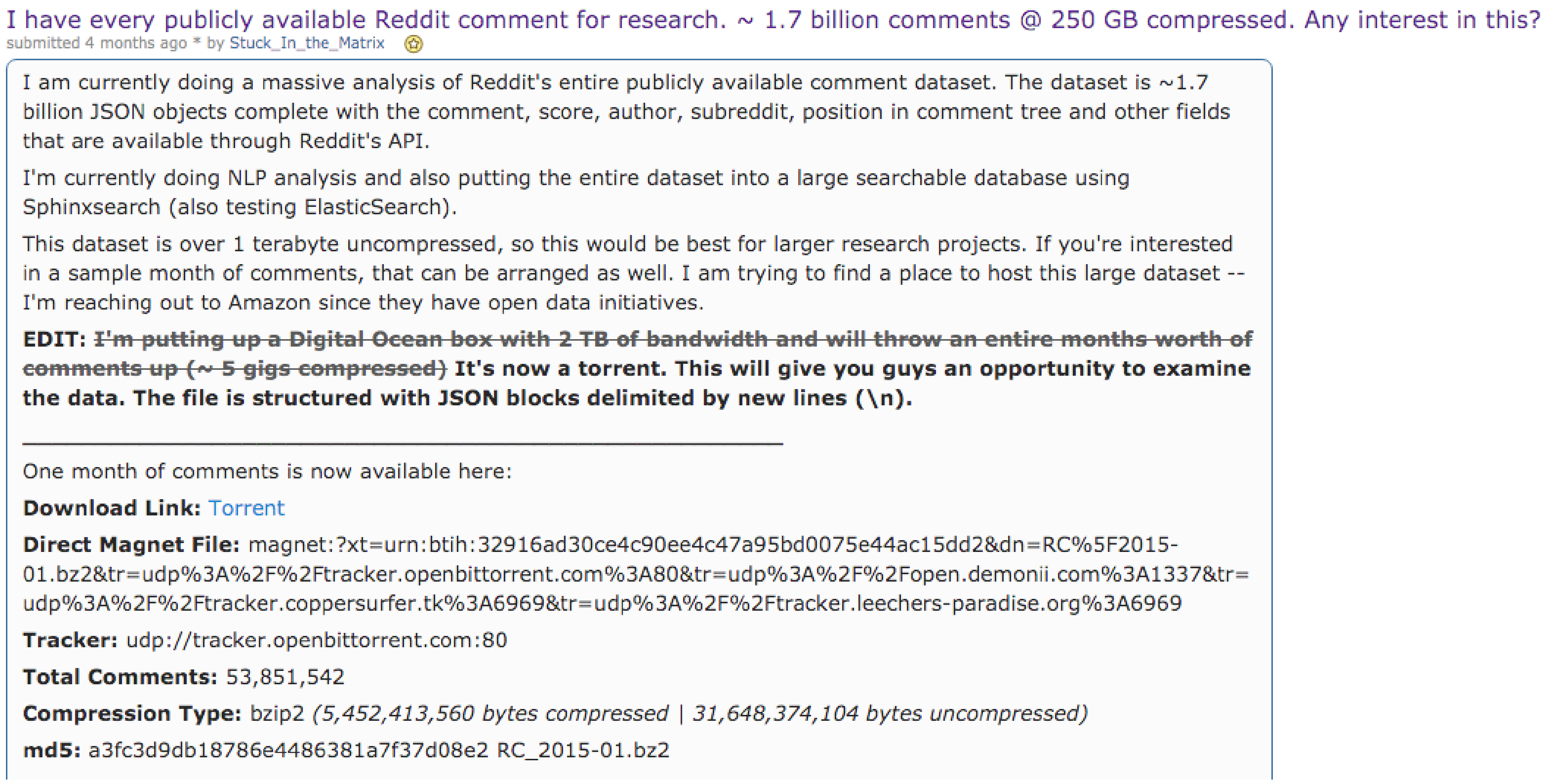
Data

reddit

The frontpage of the Internet

Data

I have every publicly available Reddit comment for research. ~ 1.7 billion comments @ 250 GB compressed. Any interest in this?
(https://www.reddit.com/r/datasets/comments/3bxlg7/i_have_every_publicly_available_reddit_co



I have every publicly available Reddit comment for research. ~ 1.7 billion comments @ 250 GB compressed. Any interest in this?
submitted 4 months ago * by Stuck_In_the_Matrix

I am currently doing a massive analysis of Reddit's entire publicly available comment dataset. The dataset is ~1.7 billion JSON objects complete with the comment, score, author, subreddit, position in comment tree and other fields that are available through Reddit's API.

I'm currently doing NLP analysis and also putting the entire dataset into a large searchable database using Sphinxsearch (also testing ElasticSearch).

This dataset is over 1 terabyte uncompressed, so this would be best for larger research projects. If you're interested in a sample month of comments, that can be arranged as well. I am trying to find a place to host this large dataset -- I'm reaching out to Amazon since they have open data initiatives.

EDIT: ~~I'm putting up a Digital Ocean box with 2 TB of bandwidth and will throw an entire months worth of comments up (~ 5 gigs compressed)~~ It's now a torrent. This will give you guys an opportunity to examine the data. The file is structured with JSON blocks delimited by new lines (\n).

One month of comments is now available here:

Download Link: Torrent

Direct Magnet File: magnet:?xt=urn:btih:32916ad30ce4c90ee4c47a95bd0075e44ac15dd2&dn=RC%5F2015-01.bz2&tr=udp%3A%2F%2Ftracker.openbittorrent.com%3A80&tr=udp%3A%2F%2Fopen.demonii.com%3A1337&tr=udp%3A%2F%2Ftracker.coppersurfer.tk%3A6969&tr=udp%3A%2F%2Ftracker.leechers-paradise.org%3A6969

Tracker: udp://tracker.openbittorrent.com:80

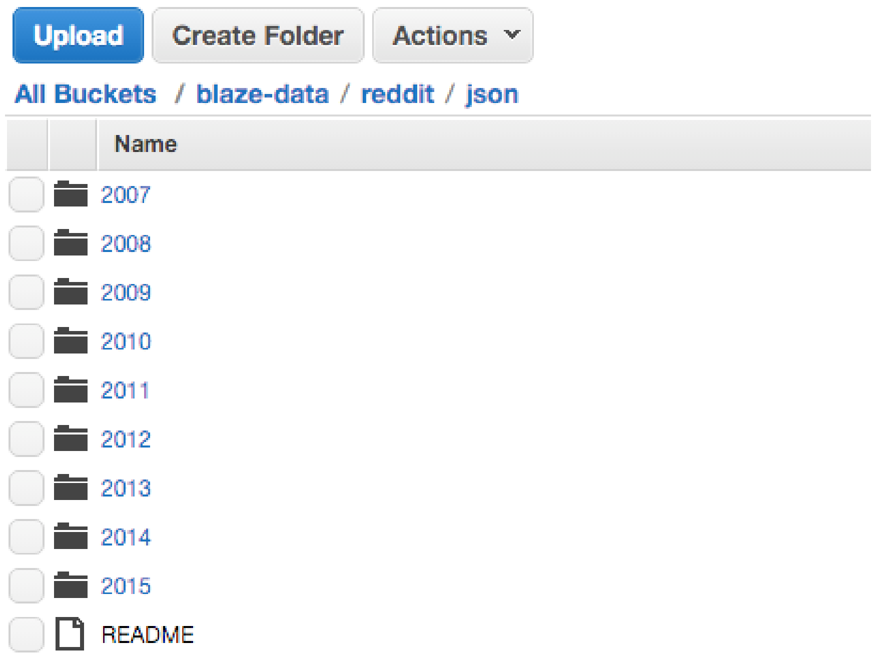
Total Comments: 53,851,542

Compression Type: bzip2 *(5,452,413,560 bytes compressed | 31,648,374,104 bytes uncompressed)*

md5: a3fc3d9db18786e4486381a7f37d08e2 RC_2015-01.bz2

Data

Is available on S3: `s3://blaze-data/reddit/json`



All Buckets / blaze-data / reddit / json

Name
2007
2008
2009
2010
2011
2012
2013
2014
2015
README

New monthly dumps at: http://pan.whatbox.ca:36975/reddit/comments/monthly/ (http://pan.whatbox.ca:36975/reddit/comments/monthly/)

Clusters

Clusters

Big Data technologies (Hadoop zoo) are here to stay

Really good management tools for IT/DevOps people from Cloudera and Hortonworks

- Automated deployment and configuration
- Customizable monitoring and reporting
- Effortless, robust troubleshooting
- Zero downtime maintenance: rolling upgrades and rollbacks
- Security: Kerberos, LDAP

Clusters

Data science "friendly" clusters

Some of the features before plus:

- Data analysis packages and environment management
- Interactive access to the cluster (Jupyter Notebook)
- Short living clusters (?)
- CLI instead of UI (?)
- More freedom (?)

Still requires to know what you are doing: AWS, Keypairs, Security groups, SSH.

No need to hide stuff. No magic.

Clusters: Anaconda Cluster

- Resource management tool that allows users to easily create, provision, and manage bare-metal or cloud-based clusters.
- It enables management of Conda environments on clusters
- Provides integration, configuration, and setup management for Hadoop
- Supported platforms include Amazon Web Services, physical machines, or even a collection of virtual machines.

http://docs.continuum.io/anaconda-cluster (http://docs.continuum.io/anaconda-cluster)

```
$ conda install anaconda-client
$ anaconda login
$ conda install anaconda-cluster -c anaconda-cluster
```

Not open source: 4 free nodes

Soon 16 free nodes in the cloud 4 in-house

Clusters: Anaconda Cluster

Provider:

```
aws:
  cloud_provider: ec2
  keyname: {{ keyname in aws }}
  location: us-east-1
  private_key: ~/.ssh/{{ keyname in aws }}.pem
  secret_id: {{ aws key }}
  secret_key: {{ aws secret }}
```

Profile:

```
name: impala-profile
provider: aws
user: ubuntu
num_nodes: 10
node_id: ami-08faa660  # Ubuntu 12.04
node_type: m3.2xlarge
root_size: 1000
plugins:
  - hdfs:
      namenode_dirs:
        - /data/dfs/nn
      datanode_dirs:
        - /data/dfs/dn
  - hive
  - impala
  - notebook
```

Launch cluster:

```
$ acluster create impala-cluster -p impala-profile
```

Clusters: Anaconda Cluster

`acluster`

```
$ acluster create name -p profile

$ acluster destroy

$ acluster ssh

$ acluster cmd 'date'

$ acluster cmd 'apt-get install build-essential' --sudo

$ acluster conda install numpy

$ acluster conda install my_pkg -c channel

$ acluster submit script.py

$ acluster put script.py /tmp/script.py

$ acluster get /tmp/script.py script.py
```

Clusters: Anaconda Cluster

`acluster install`

```
$ acluster install hdfs
$ acluster install hive
$ acluster install impala

$ acluster install elasticsearch
$ acluster install kibana
$ acluster install logstash

$ acluster install notebook
$ acluster install spark-standalone
$ acluster install spark-yarn

$ acluster install storm

$ acluster install ganglia
```

Clusters: Anaconda Cluster

```
$ acluster conda install -c r r-essentials
Installing packages on cluster "impala": r-essentials

Node "ip-172-31-0-186.ec2.internal":
    Successful actions: 1/1
Node "ip-172-31-0-190.ec2.internal":
    Successful actions: 1/1
Node "ip-172-31-0-182.ec2.internal":
    Successful actions: 1/1
Node "ip-172-31-0-189.ec2.internal":
    Successful actions: 1/1
Node "ip-172-31-0-191.ec2.internal":
    Successful actions: 1/1
Node "ip-172-31-0-183.ec2.internal":
    Successful actions: 1/1
Node "ip-172-31-0-184.ec2.internal":
    Successful actions: 1/1
Node "ip-172-31-0-187.ec2.internal":
    Successful actions: 1/1
Node "ip-172-31-0-185.ec2.internal":
    Successful actions: 1/1
Node "ip-172-31-0-188.ec2.internal":
    Successful actions: 1/1
```

Clusters: DataScienceBox

Pre Anaconda Cluster: Command line utility to create instances in the cloud ready for data science. Includes conda package management plus some Big Data frameworks (spark).

https://github.com/danielfrg/datasciencebox (https://github.com/danielfrg/datasciencebox)

```
$ pip install datasciencebox
```

CLI will be available:

```
$ datasciencebox
$ dsb

$ dsb up

$ dsb install miniconda
$ dsb install conda numpy

$ dsb install notebook
$ dsb install hdfs
$ dsb install spark
$ dsb install impala
```

Clusters: Under the Hood

Both DSB and AC use a very similar approach: SSH for basic stuff and then use Salt

SALTSTACK

Salt: https://github.com/saltstack/salt (https://github.com/saltstack/salt)

- 100% free and open source
- Fast: ZMQ instead of SSH
- Secure
- Scalable to thousands of nodes
- Declarative yaml languague instead of bash scripts

```
numpy-install:
  conda.installed:
    - name: numpy
    - require:
        - file: config-file
```

- A lot of free formulas online

Data

Data: Moving the data

Move data from S3 to our HDFS cluster

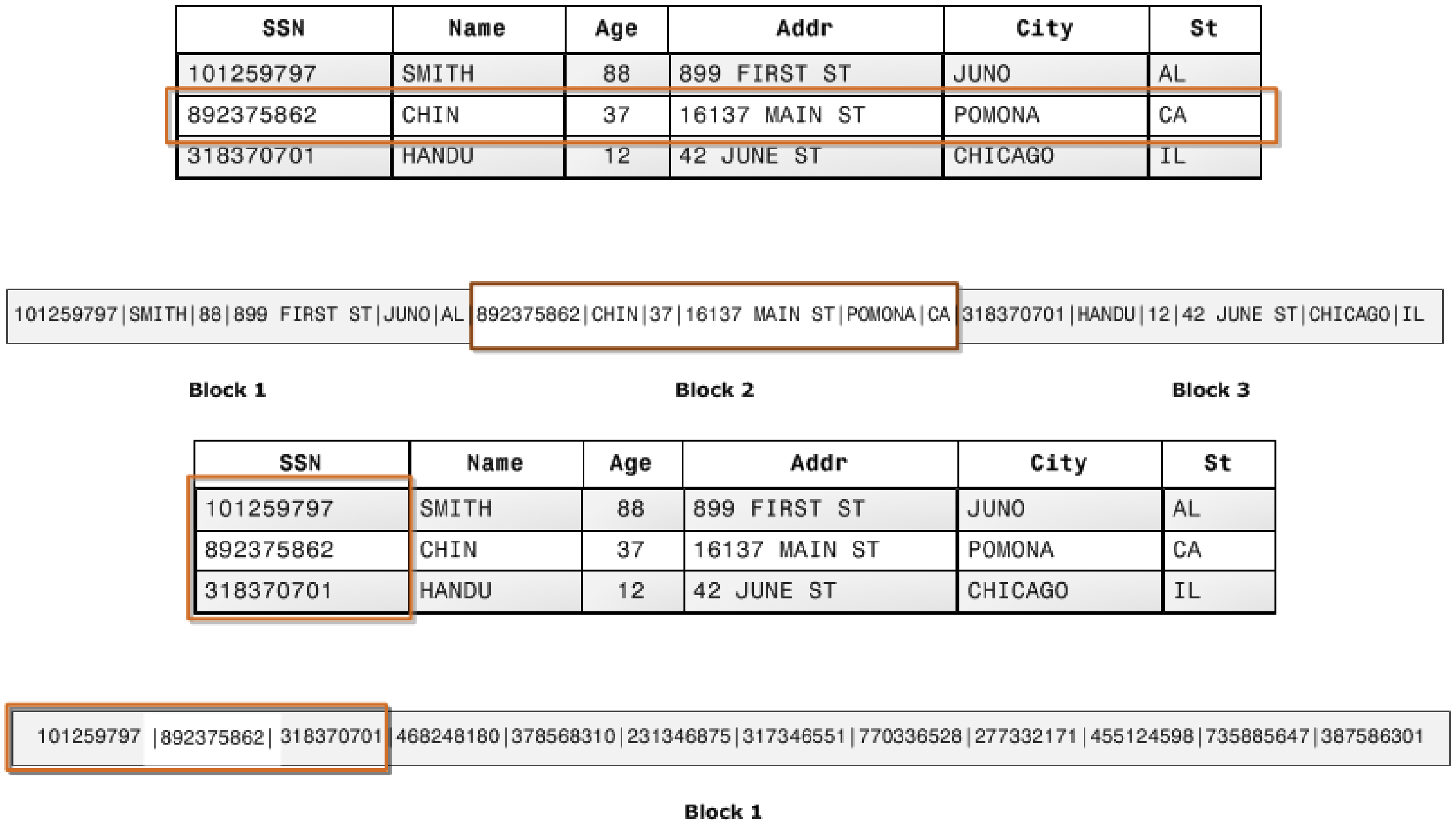
```
hadoop distcp -Dfs.s3n.awsAccessKeyId={{ }} -Dfs.s3n.awsSecretAccessKey={{ }}
s3n://blaze-data/reddit/json/*/*.json /user/ubuntu
```

Data: Parquet

Parquet

Apache Parquet is a columnar storage format available to any project in the Hadoop ecosystem, regardless of the choice of data processing framework, data model or programming language.

Data: Parquet



SSN	Name	Age	Addr	City	St
101259797	SMITH	88	899 FIRST ST	JUNO	AL
892375862	CHIN	37	16137 MAIN ST	POMONA	CA
318370701	HANDU	12	42 JUNE ST	CHICAGO	IL

```
101259797|SMITH|88|899 FIRST ST|JUNO|AL|892375862|CHIN|37|16137 MAIN ST|POMONA|CA|318370701|HANDU|12|42 JUNE ST|CHICAGO|IL
```

SSN	Name	Age	Addr	City	St
101259797	SMITH	88	899 FIRST ST	JUNO	AL
892375862	CHIN	37	16137 MAIN ST	POMONA	CA
318370701	HANDU	12	42 JUNE ST	CHICAGO	IL

```
101259797 |892375862| 318370701|468248180|378568310|231346875|317346551|770336528|277332171|455124598|735885647|387586301
```

[1] http://docs.aws.amazon.com/redshift/latest/dg/c_columnar_storage_disk_mem_mgmnt.html

Data: Load

```
hive > CREATE TABLE reddit_json (
  archived                   boolean,
  author                     string,
  author_flair_css_class     string,
  author_flair_text          string,
  body                       string,
  controversiality           int,
  created_utc                string,
  distinguished              string,
  downs                      int,
  edited                     boolean,
  gilded                     int,
  id                         string,
  link_id                    string,
  name                       string,
  parent_id                  string,
  removal_reason             string,
  retrieved_on               timestamp,
  score                      int,
  score_hidden               boolean,
  subreddit                  string,
  subreddit_id               string,
  ups                        int
)
ROW FORMAT
    serde 'com.amazon.elasticmapreduce.JsonSerde'
    with serdeproperties ('paths'='archived,author,author_flair_css_class,author_flai
r_text,body,controversiality,created_utc,distinguished,downs,edited,gilded,id,link_i
d,name,parent_id,removal_reason,retrieved_on,score,score_hidden,subreddit,subreddit_i
d,ups');


hive > LOAD DATA INPATH '/user/ubuntu/*.json' INTO TABLE reddit_json;
```

Data: Transform

```
hive > CREATE TABLE reddit_parquet (
  archived                 boolean,
  author                   string,
  author_flair_css_class   string,
  author_flair_text        string,
  body                     string,
  controversiality         int,
  created_utc              string,
  distinguished            string,
  downs                    int,
  edited                   boolean,
  gilded                   int,
  id                       string,
  link_id                  string,
  name                     string,
  parent_id                string,
  removal_reason           string,
  retrieved_on             timestamp,
  score                    int,
  score_hidden             boolean,
  subreddit                string,
  subreddit_id             string,
  ups                      int,
  created_utc_t            timestamp
)
STORED AS PARQUET;


hive > SET dfs.block.size=1g;

hive > INSERT OVERWRITE TABLE reddit_parquet select *, cast(cast(created_utc as doubl
e) as timestamp) as created_utc_t FROM reddit_json;
```

Querying

Querying

Using the regular hive/impala shell

```
impala > invalidate metadata;

impala > SELECT count(*) FROM reddit_parquet;
Query: select count(*) FROM reddit_parquet
+------------+
| count(*)   |
+------------+
| 1830807828 |
+------------+

Fetched 1 row(s) in 4.88s
```

Querying with python

Numpy and Pandas like API that targets not local files but another engines

SQL:

- Postgres
- Impala
- Hive
- Spark SQL

NoSQL:

- Mongo DB
- Still local files

Projects:

- Blaze: github.com/blaze/blaze (https://github.com/blaze/blaze)
- Ibis: github.com/cloudera/ibis (https://github.com/cloudera/ibis)

Blaze

An interface to query data on different storage systems

Code at https://github.com/blaze/blaze (https://github.com/blaze/blaze)

Blaze ecosystem: http://blaze.pydata.org (http://blaze.pydata.org)

```
In [1]: import blaze as bz
        import pandas as pd
```

```
In [2]: data = bz.Data('impala://54.209.0.148/default::reddit_parquet')
```

Blaze

Number of comments

```
In [4]: data.id.count()
```

```
Out[4]: 1830807828
```

```
In [5]: print(bz.compute(data.id.count()))
```

```
SELECT count(reddit_parquet.id) AS id_count
FROM reddit_parquet
```

Blaze

Total number of up votes

```
In [6]: n_up_votes = data.ups.sum()
```

```
In [7]: print(bz.compute(n_up_votes))
```

```
SELECT sum(reddit_parquet.ups) AS ups_sum
FROM reddit_parquet
```

```
In [8]: %time int(n_up_votes)
```

```
CPU times: user 22.4 ms, sys: 6.84 ms, total: 29.2 ms
Wall time: 3.69 s
```

```
Out[8]: 9696701385
```

Blaze

Counting the total number of posts in the `/r/soccer` subreddit

```
In [9]: n_posts_in_r_soccer = data[data.subreddit == 'soccer'].id.count()
```

```
In [10]: print(bz.compute(n_posts_in_r_soccer))
```

```
SELECT count(alias_1.id) AS id_count
FROM (SELECT reddit_parquet.id AS id
FROM reddit_parquet
WHERE reddit_parquet.subreddit = %(subreddit_1)s) AS alias_1
```

```
In [11]: %time int(n_posts_in_r_soccer)
```

```
CPU times: user 28.6 ms, sys: 8.61 ms, total: 37.2 ms
Wall time: 5 s
```

```
Out[11]: 13078620
```

Blaze

Counting the number of comments before a specific hour

```
In [12]: before_1pm = data.id[bz.hour(data.created_utc_t) < 13].count()
```

```
In [13]: print(bz.compute(before_1pm))
```

```
SELECT count(alias_3.id) AS id_count
FROM (SELECT reddit_parquet.id AS id
FROM reddit_parquet
WHERE EXTRACT(hour FROM reddit_parquet.created_utc_t) < %(param_1)s) AS alias_3
```

```
In [14]: %time int(before_1pm)
```

```
CPU times: user 32.7 ms, sys: 9.88 ms, total: 42.6 ms
Wall time: 5.54 s
```

```
Out[14]: 812870494
```

Blaze

Plotting the daily frequency of comments in the `/r/IAmA` subreddit

```
In [15]: iama = data[(data.subreddit == 'IAmA')]
```

```
In [16]: days = (bz.year(iama.created_utc_t) - 2007) * 365 + (bz.month(iama.created_utc_t) - 1)
         * 31  + bz.day(iama.created_utc_t)
```

```
In [17]: iama_with_day = bz.transform(iama, day=days)
```

```
In [18]: by_day = bz.by(iama_with_day.day, posts=iama_with_day.created_utc_t.count())
```

Blaze

Plotting the daily frequency of comments in the `/r/IAmA` subreddit

Pandas

```
In [19]: by_day_result = bz.odo(by_day, pd.DataFrame)  # Actually triggers the computation
```

```
In [20]: by_day_result.head()
```

Out[20]:

	day	posts
0	2405	16202
1	2978	2361
2	1418	5444
3	1874	8833
4	1257	4480

```
In [21]: by_day_result = by_day_result.sort_values(by=['day'])
```

```
In [22]: rng = pd.date_range('5/28/2009', periods=len(by_day_result), freq='D')
         by_day_result.index = rng
```

Blaze

Plotting the daily frequency of comments in the `/r/IAmA` subreddit

```
In [23]: from bokeh._legacy_charts import TimeSeries, output_notebook, show
```

```
In [24]: output_notebook()
```

(http://bokeh.pydata.org)
BokehJS successfully loaded.

Ibis

Ibis is a new Python data analysis framework with the goal of enabling data scientists and data engineers to be as productive working with big data as they are working with small and medium data today. In doing so, we will enable Python to become a true first-class language for Apache Hadoop, without compromises in functionality, usability, or performance

Code at:

More info: http://www.ibis-project.org (http://www.ibis-project.org)

Ibis

Number of posts with more than 1k up votes

In [1]:
```
import ibis
from ibis.impala.compiler import to_sql

import pandas as pd
```

In [2]:
```
ibis.options.interactive = True
ibis.options.sql.default_limit = 20000
```

In [3]:
```
hdfs = ibis.hdfs_connect(host='52.91.39.64')
con = ibis.impala.connect(host='54.208.255.126', hdfs_client=hdfs)
```

In [4]:
```
data = con.table('reddit_parquet')
```

Ibis

```
In [5]: data.schema()
```

```
Out[5]: ibis.Schema {
          archived                  boolean
          author                    string
          author_flair_css_class    string
          author_flair_text         string
          body                      string
          controversiality          int32
          created_utc               string
          distinguished             string
          downs                     int32
          edited                    boolean
          gilded                    int32
          id                        string
          link_id                   string
          name                      string
          parent_id                 string
          removal_reason            string
          retrieved_on              timestamp
          score                     int32
          score_hidden              boolean
          subreddit                 string
          subreddit_id              string
          ups                       int32
          created_utc_t             timestamp
        }
```

Ibis

Number of posts with more than 1k up votes

```
In [6]: more_than_1k = data[data.ups >= 1000]
```

```
In [7]: month = (more_than_1k.created_utc_t.year() - 2007) * 12 + more_than_1k.created_utc_t.mo
        nth()
        month = month.name('month')
```

```
In [8]: with_month = more_than_1k['id', month]
```

```
In [9]: posts = with_month.count()
        groups = with_month.aggregate([posts], by='month')
```

```
In [10]: month_df = groups.execute()
```

Ibis

Number of posts with more than 1k up votes

Pandas

In [11]:
```
month_df = month_df.set_index('month')
month_df.sort_index(inplace=True)
rng = pd.date_range('10/01/2007', periods=len(month_df), freq='M')
month_df.index = rng
```

In [12]:
```
from bokeh._legacy_charts import TimeSeries, output_notebook, show

output_notebook()
```

(http://bokeh.pydata.org)
BokehJS successfully loaded.

Spark?

Data is on HDFS in Parquet, Spark is happy with that

```
In [ ]: from pyspark.sql import SQLContext
        sqlContext = SQLContext(sc)
```

```
In [ ]: # Read in the Parquet file
        parquetFile = sqlContext.read.parquet("people.parquet")

        # Parquet files can also be registered as tables and then used in SQL statements
        parquetFile.registerTempTable("parquetFile");
        teenagers = sqlContext.sql("SELECT name FROM parquetFile WHERE age >= 13 AND age <= 1
        9")

        # These are spark DataFrames so you can do other stuff like map
        teenNames = teenagers.map(lambda p: "Name: " + p.name)
        for teenName in teenNames.collect():
            print(teenName)
```

UDFs

Taken from: http://spark.apache.org/docs/latest/sql-programming-guide.html#parquet-files

Wrap up

- We can make Data Scientist access to big data tools easier
- Data Scientists need to understand the underlying big data and dev ops tools to some degree
- Some of these tools are very useful for SQL type queries (BI, Tableau) but for more advanced analytics or ML other tools are needed
- Download Anaconda Cluster (or DataScienceBox) and try for yourself

Ideas:

Thanks