The Billion Object Platform (BOP): A Real-time, Big Data, Spatio-Temporal Exploration Platform

Harvard ABCD-GIS

Benjamin Lewis, Merce Crosas, David Smiley, Devika Kakkar, Ariel Nunez
Outline

• Introduction
• Architecture
• Harvesting/ Archiving
• Sentiment Enrichment
• Apache Kafka
• Solr for Geo-enrichment
• Solr & Time Sharding
• BOP Web-Service
• Client UI
• Deployment/ Operations
• Docker and Kontena
BOP Requirements Summary

Provide a proof-of-concept platform designed to lower the barrier for researchers who need to access big streaming spatio-temporal datasets.

- Most recent ~billion geo-tweets
- Realtime search (<5 sec latency)
- Sub-second queries
  - Including heatmaps!
- On the cheap: ~6 commodity servers
BOP as an Example of a New Kind of Dataset Available in Dataverse
Streaming Data: Harvest and Archive

Latest 1 Billion GeoTweets
- "Live system" / Solr
- Instant search results
- Easy exploration
- Approx. 3 months of data

Archived: Billions of GeoTweets
- Current: Manual access
- Future
- Automated searches
- Reproduce data subsets built/discovered in the "live system"
Initial focus on Geo-tweets
(could be any streaming dataset)

• 1-2% of tweets have GPS coordinates from the user’s device, ranges from
1 to 6 million per day
• The CGA has been harvesting geo-tweets since 2012 and has an informal
archive of about 8 billion objects
• Researcher Ryan Qi Wang also harvesting during this period. His tweets
were loaded first. CGA tweets will be merged later.
• Collaborators:
  • Harvard Dataverse Team
  • Boston Area Research Initiative
Logical High-Level Architecture

Harvesting → Enrichment → Kafka (archive) → Solr → Web Service → Browser UI

Data flows via Apache Kafka

Docker, Kontena, OpenStack

Hosting: Mass OpenCloud
Apache Kafka

- Kafka: a scalable message/queue platform
- See new Kafka Streams & Kafka Connect APIs
- No back-pressure; can be a challenge
- Non-obvious use:
  - For storage; time partitioning
    - Lots of benefits yet serious limitations
Real-Time Harvesting

Connect to Twitter’s Streaming API

Stream tweets using predefined users and coordinates extent

If the tweet is Geotagged

Kafka Topic
Enrichment

Geo: Query Solr via spatial point query; attach related metadata to tweet
Sentiment Analysis

- Classifier: Support Vector Machine (SVM) with Linear Kernel
- Source code in Python
- Uses scikit-learn, numpy, scipy, NLTK
- Two classes of sentiment: Positive (1), Negative (0)
- Training Corpus: Sentiment140, Polarity dataset v2.0, University of Michigan
- Preprocessing: Lower case, URLs, @user, #tags, trimming, repeating characters, emoticons
- Stemming: Porter stemmer
- Precision, Recall, F1 score: 0.82 (82%)
- Processing speed: 20ms/tweet (no emoticon), 5ms/tweet (emoticon)
Sentiment Analysis

Phase 1: Training

Train the classifier → Save as pickle

Phase 2: Prediction

Load the classifier → Parse → Preprocess → Stem → Predict

For each tweet
Solr for Geo Enrichment

“Reverse Geocoding”

• Tweets (docs) can have a geo lat/lon
• Enrich tweet with Country, State/Province, ...
  – Gazetteer lookup (point-in-polygon)

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Features</th>
<th>Raw size</th>
<th>Index time</th>
<th>Index size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin2</td>
<td>46,311</td>
<td>824 MB</td>
<td>510 min</td>
<td>892 MB</td>
</tr>
<tr>
<td>US States</td>
<td>74,002</td>
<td>747 MB</td>
<td>4.9 min</td>
<td>840 MB</td>
</tr>
<tr>
<td>Massachusetts Census Blocks</td>
<td>154,621</td>
<td>152 MB</td>
<td>5.9 min</td>
<td>507 MB</td>
</tr>
</tbody>
</table>
Fast Point-in-Polygon Tricks

Index/Config

- Optimize to 1 segment
- RptWithGeometry SpatialField
  - precisionModel="floating_single"
  - autoIndex="true"
- <cache name="perSegSpatial FieldCache_WKT" ...>

Search

- Embed Solr (in-process)
- Use docValues, not stored
  - fl=block:field(GEOID10)

Query like this:

- q={!field cache=false f=WKT}Intersects(POINT($lon $lat))
Apache Solr

- Search / analytics server, based on Lucene
- Custom add-ons:
  - Time sharded routing (index + query)
  - LatLonPointSpatialField – in Solr 6.5
    - Faster/leaner search & sort for point data
  - HeatmapSpatialField – in Solr 6.6 TBD
    - Faster/leaner heatmaps at scale
Time “Sharding”

Solr has no built-in time based sharding.

A Solr custom “URP” was developed to route tweets to the right by-month shard. It auto creates and deletes shards.

A Solr custom “SearchHandler” was developed to decide which subset of shards to search based on custom parameters sent by the web-service.

Generally useful for others. Need more work for contribution to Solr itself.
The BOP Web-Service

- HTTP/REST API
  - Keyword search
  - Faceting
    - Heatmaps
  - CSV export
- Why not Solr direct?
  - Define a supported API
  - Ease of use for clients
  - Security

Tech:
- Swagger
- Dropwizard
- Kotlin lang (on JVM)
Client UI

• Browser side UI with no server component
• It uses the following technologies:
  – Angular JS
  – OpenLayers 3
  – npm (dependencies, script minification, development)
UI adapts to laptop
UI adapts to tablets
UI adapts to phones
Temporal filtering
Temporal faceting (histogram)
Spatial filtering
Spatial faceting (heatmap)
Text faceting (tag cloud)
Nearby tweets
Deployment / Operations

• MassOpenCloud “MOC”
  – OpenStack based cloud (mimics Amazon EC2)

• CoreOS

• Kontena & Docker

• Admin/Ops tools:
  – Kafka Manager (Yahoo!)
  – Solr’s admin UI

Stats:
• 12 nodes (machines)
  • 5 to Solr
  • 3 to Kafka
  • 3 to enrichment, …
• 217 GB RAM
• 3500 GB disk
• 17 services (software pieces)
  • 133 containers
Docker

- Easy to find/try/use software
  - No installation
  - Simplified configuration (env variables)
  - Common logging
  - Isolated

- Ideal for:
  - Continuous Int. servers
  - Trying new software
  - Production advantages too
    - but “new”
Docker in Production

• We use “Kontena”
• Common logging, machine/proc stats, security
  – VPN to secure network; access everything as local
• No longer need to care about:
  – Ansible, Chef, Puppet, etc.
  – Security at network or proxy; not service specific
• Challenges: state & big-data
Thank you

David Smiley
david.w.smiley@gmail.com

Devika Kakkar
kakkar@fas.harvard.edu