The Billion Object Platform (BOP): a system to lower barriers to support big, streaming, spatio-temporal data sources

Devika Kakkar and Ben Lewis
Harvard Center for Geographic Analysis

David Smiley
Independent consultant

Ariel Nunez
Terranodo
Goal

Develop a platform that makes it easier for researchers to interactively explore large spatio-temporal datasets.
Initial focus on geo-tweets
(but could be any streaming dataset)

• 1-2% of tweets have GPS coordinates from the user’s device, currently about 1 million per day available via the Twitter API
• The CGA has been harvesting geo-tweets since 2012 and has an informal archive of about 8 billion objects
• Northeastern Professor Ryan Qi Wang also harvested during this period and we plan to eventually merge the two datasets to create a more complete version.
Requirements

• Develop back end and client to support interactive visualization of a billion point features
• Support sub-second queries including heatmaps and temporal histograms
• Expose a general purpose RESTful API
• Run system on low cost commodity hardware or VMs
Latest billion + long term archive

Latest ~billion geo-tweets
- Sharded Solr/Lucene
- Streaming update
- Interactive exploration
- Approx 3 years of data

Archive for persistent storage all tweets
- Currently on Kafka
- May move to Swift
- Eventual access from Cloud Dataverse
- Back to 2012
Note:
BOP visualization (2D faceting) originally developed for the HHypermap Registry
BOP demo URL
https://youtu.be/tib6M_fgHok
Logical High-Level Architecture

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

Data flows via **Apache Kafka**

Hosting: Mass OpenCloud

BOP

Docker, Kontena, OpenStack
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

1. Connect to Twitter’s Streaming API
2. Stream tweets using predefined users and coordinates extent
3. If the tweet is Geotagged
4. 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**
- For each tweet
  - Load the classifier
  - Parse
  - Preprocess
  - Stem
  - Predict
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>
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
    • 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.
Solr Heatmaps: Grid Faceting

- Spatial density summary grid faceting, also useful for point-plotting search results
- Lucene & Solr APIs
- Scalable & fast *usually...*
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)
UI Stack

• BOP’s UI uses the following technologies:
  – Angular JS
  – OpenLayers 3
  – npm (dependencies, script minification, development)
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
Next steps

- Persistent archive in Swift object storage
- Exploring adding analytic capabilities using GeoMesa
- Support faceting on numeric values (in addition to counts) to support other types of visualizations.
Thank you

Ben Lewis
blewis@cga.harvard.edu

David Smiley
david.w.smiley@gmail.com

Devika Kakkar
kakkar@fas.harvard.edu
Backup slides
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