BUILDING A BILLION SPATIO-TEMPORAL OBJECT SEARCH AND VISUALIZATION PLATFORM

D. Kakkar [a], B.Lewis [a]

[a] Center for Geographic Analysis, Harvard University, Cambridge, MA, USA – kakkar@fas.harvard.edu, blewis@cga.harvard.edu

KEY WORDS: Solr, Lucene, big data, OpenStack, containers, visualization, social media, spatio-temporal, open source, research, GIS, geospatial, spatio-temporal, real time, streaming, cloud-based

ABSTRACT:

With funding from the Sloan Foundation and Harvard Dataverse, the Harvard Center for Geographic Analysis (CGA) has developed a prototype spatio-temporal visualization platform called the Billion Object Platform or BOP. The goal of the project is to lower barriers for scholars who wish to access large, streaming, spatio-temporal datasets. The BOP is now loaded with the latest billion geo-tweets, and is fed a real-time stream of about 1 million tweets per day. The geo-tweets are enriched with sentiment and census/admin boundary codes when they enter the system. The system is open source and is currently hosted on Massachusetts Open Cloud (MOC), an OpenStack environment with all components deployed in Docker orchestrated by Kontena. This paper will provide an overview of the BOP architecture, which is built on an open source stack consisting of Apache Lucene, Solr, Kafka, Zookeeper, Swagger, scikit-learn, OpenLayers, and AngularJS. The paper will further discuss the approach used for harvesting, enriching, streaming, storing, indexing, visualizing and querying a billion streaming geo-tweets**.**

1. INTRODUCTION

1.1 Background

The Billion Object Platform (BOP) is a prototype platform designed to lower the barrier for researchers who need to access big streaming spatio-temporal datasets. This system addresses the need for a way to manage real-time streaming flows of big-data, and presents a prototype specifically aimed at storing, indexing, visualizing and querying geo-tweets. The system exposes the latest billion raw objects for filtering by basic dimension: time, space, keyword and supports interactive visualization of distributions in time and space. Geo-tweets are tweets containing a GPS coordinate from the originating device. About 1-2% of all tweets are geo-tweets.. The Centre for Geographic Analysis (CGA) at Harvard has been harvesting geo-tweets since 2012 and has an archive of about 8 billion objects to date. The BOP builds on the 2D grid heatmap faceting capability of Apache Lucene/Solr, which the CGA developed with funding from the National Endowment for the Humanities. This 2D same faceting technology is also used by a map service discovery platform developed at CGA called HHypermap Registry.

1.2 Functional Requirements

The goal of BOP is to develop a platform to lower barriers to the access of large streaming datasets. The main functional requirements of the system are:

1. Provide access to the most recent ~billion geo-tweets
2. Real-time search (<5 seconds)
3. Sub-second queries including heatmaps and temporal histograms
4. On the cheap using commodity servers

The BOP provides access to a new kind of dataset within Harvard Dataverse, exposing an API that can be plugged into Dataverse and enable subsets to be pushed into Dataverse for further analysis. This instance of the BOP system is loaded with geo-referenced tweets but a similar dataset could, with some tweaking, be substituted.

1.3 System Architecture

The high-level architecture of the BOP is shown in Figure 1. The primary infrastructure components are Apache Kafka and Apache Solr. Harvard CGA is harvesting geotweets, which are then enriched with metadata in the form of sentiment analysis and spatial joins with census and other boundary datasets. All geo-tweets are archived using a long-term Kafka topic. The BOP itself, which consists of a Solr index–based copy of the data, represents the latest billion while the index goes back further in time. The BOP-core exposes a search and extraction web service API that the client consumes. All components are deployed to a Docker based infrastructure managed by Kontena. The term "BOP-core" is used to refer to both the Solr index and the web service that exposes it.

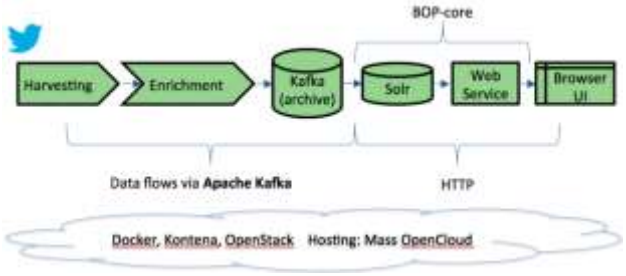

Figure 1. Logical high-level architecture of BOP

1.4 Preliminary System Performance

The performance of the system is difficult to characterize, as there are many variables at play: the document size of the largest Solr shard being queried, the number of docs matching a query in that shard, whether the same query has been run before

*Corresponding author

(thus cached), the nature of the filters (keyword, space, time, others), and the optional components of a query: requesting top docs, requesting heatmaps (at given resolution), requesting temporal histograms (at given resolution), and whether or not the leading shard (for the current time period) is queried. The table of times below has two columns; both were for separate 12 month periods (across 12 Solr shards); the first in 2016, the second in 2015. We have more data in 2015 than in 2016. The document count of the largest shard (month) is indicated as it has a large effect. Times are in milliseconds.

	Largest Shard	
	17.8M docs	139M docs
filters (F)	1100	3110
F and top docs	800	930
F and heatmaps	65	365
F and histogram	22	25

Table 1. Performance Results

The first row is the cold (not-cached) time it took to return a simple document count for the keyword query "usa" along with the temporal period for that year, and a spatial filter covering the USA, northeast region. The temporal and spatial aspects were cached but the keyword was chosen anew. The performance of this can vary a great deal. If the query has been issued before, this time can vanish. The second row is the cold (not cached) time for returning the 25 top docs (sorted by time). Again this can vary a lot. If the query has been issued before, this time can vanish. The third row is the time to compute a heatmap of 6003 cells. This is not sensitive to caching because it is not cached.The fourth row is the time to compute the temporal histogram; for a time range of 53 weeks. The performance of this and the sensitivity to caching depends on the presence of filters. It can be much slower for queries lacking a keyword filter.

1.5 Enhancements to core Solr and Lucene

Solr is an enterprise search server based on Apache Lucene. The following enhancements were made to Solr for the purpose of this project:

1. Time sharded routing for indexing and querying
2. LatLonPointSpatialField – in Solr 6.5 for faster/ leaner search of point data
3. HeatmapSpatialField – in Solr 6.5 for faster/leaner heatmaps at scale

2. COMPONENTS

2.1 Harvesting/Archiving

Harvesting is the process of capturing real-time data and getting it it loaded to the BOP. The harvesting service continuously polls Twitter's API based on predefined users and coordinate extents and then deposits the JSON tweets into a Kafka topic. We serialize the tweet JSON using MessagePack. The latest billion tweets are then pushed to Solr and are made available for instant search and easy exploration within BOP.The harvested data is also stored in a long-term archive. CGA has been harvesting tweets since 2012 and has archive of about 8 billion objects to date.

2.2 Enrichment

Enrichment consists of binary sentiment analysis (e.g. happy/sad) and geo enrichment (e.g. USA/ Massachusetts) — reverse-geocoding tagging. Both these enrichments will be discussed in detail in sections below. As seen in Figure 2, this service consumes an input Kafka topic, and then adds the enriched tweets to another topic.

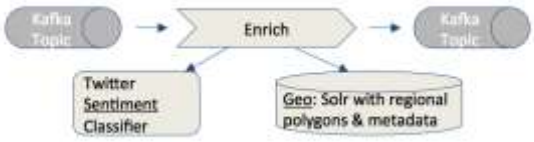

Figure 2. Process for enrichment in BOP

2.2.1 Sentiment Analysis

Sentiment analysis is a field of study, which attempts to identify the emotion expressed in text using natural language processing (NLP) tools. Social media platforms such as Twitter provide a constant source of textual data, much of which is contains an emotion, which can be used to help determine sentiment using Sentiment Analysis tools. We use the Support Vector Machine (SVM) algorithm with a linear kernel, which is written in python and uses the scikit-learn library. Two classes of sentiment are assigned: positive (1) and negative (0). Tweets are pre-processed before sentiment stamping. The training corpi used are Stanford's Sentiment140, Polarity dataset v2.0, and the University of Michigan's SI650-sentiment classification dataset. The overall precision, recall and F1-score of the algorithm achieved is 82%. Sentiment enrichment takes on an average 20 milliseconds for a tweet with no emoticon and 5 milliseconds for a tweet with an emoticon.

2.2.2 Solr for Geo-enrichment

Geo-enrichment is basically reverse geocoding where we use the geospatial coordinates for a tweet to check which census or county or country administrative polygon it falls within. This project specially configured Solr to support very fast point in polygon lookups. The total number of polygons used in our enrichment is about 250,000. The three queries for reverse geocoding enrichment take sub-milliseconds each.

2.2.3 Removing Selected Bots

A twitter bot is a software program that sends out automated posts with auto-generated geospatial coordinates via Twitter. These seemingly randomly located tweets spread across oceans and the globe add static to the geo-visualization without providing meaningful content in return. Therefor we remove such bots from the system as we find them, and document the bots we have removed so users can retrieve them if needed using the Twitter API.

2.3 Apache Kafka

Apache Kafka is a scalable message\queue platform designed to handle high data volumes. The latest Kafka Streams API was found to be very useful for consuming from a stream and then passing the content to another stream. In this project, the API was also used for moving data from Kafka into other systems like Solr. Kafka does not support backpressure mechanisms,

which creates challenges in some cases like using Kafka to re-index everything. A non- obvious use of Kafka we implemented in this project is that of a persistent, long-term data store or archive. We used long-term topics in Kafka to accomplish this. This approach for archiving has advantages but also some serious limitations such as the fact data cannot be sorted or de-duplicated within Kafka as it could be in a database.

2.4 Solr and Time Sharding

Tweets are brought into to Solr and indexed by a service called Ingest, which consumes from a Kafka topic, then maps Tweet JSON to a SolrInputDocument that aligns with our schema, and then sends it to Solr. The Solr configuration has two parts: (A) the Solr collection to send the document to, and (B) a "solrConnectionString" which might either be an HTTP reference to a Solr node, or SolrCloud zookeeper coordinates. Solr has no built-in time-based sharding, so a Solr custom Update Request Processor (URP) was developed to route tweets to the right by-month shard. The URP also auto creates and deletes shards. Related to this we also developed a custom Solr Search Handler that looks for special start-end parameters to determine which date shards to route the request to.

2.5 BOP Web-Service

We developed a RESTful API called the BOP Webservice to support our BOP client and potentially other clients. The web service supports keyword search, space and time-based faceting, and CSV export. Having our own API instead of using Solr directly (which would also be possible) makes it much easier for developers to use, plus it adds security. The BOP API is based on API frameworks Swagger and Dropwizard, and is written in the Kotlin language.

2.6 Client UI

The BOP client shown in Figure 3 is a browser-based UI with no server component, which makes HTTP/REST requests to the BOP Webservice. HTTP requests are proxied through a Kontena loadbalancer to provide access to the Kontena "weave" network which is one of the first overlay network technologies made available for Docker.

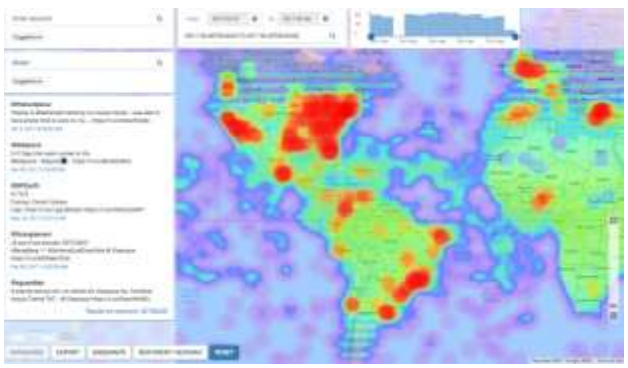

Figure 3. BOP Client UI

The client UI is build using Angular JS, OpenLayers3, node packet manager.
The client UI can adapt to computers, tablets and phones and offers the following functionalities:

1. Temporal filtering
2. Temporal faceting (Histogram)
3. Spatial filtering
4. Spatial faceting (Heatmaps)
5. Text faceting (Tag cloud)

The client allows downloads of up to 10,000 tweet IDs along with the enrichment fields in a CSV format. A tool has been developed for the "rehydration" of these tweet IDs which involves sign the Twitter API to to fetch full tweet JSON..

3. DEPLOYMENT/ OPERATION

3.1 Massachusetts Open Cloud (MOC)

The BOP system is hosted on servers run by Massachusetts Open Cloud (MOC) which is a public cloud based on Open Cloud eXchange model built on OpenStack software. The system runs on 12 virtual server instances: 5 for Solr, 3 for Kafka, 3 for enrichment, 1 for bop-pub. It uses 217GB RAM, 3500 GB of disk space, and consists of 17 services (133 containers).

3.2 Docker/ Kontena

All components of BOP are deployed to a Docker based infrastructure managed by Kontena. Docker is a leading software container platform, which provides ease of use: no installation, simplified configurations using environment variables and common logging mechanism. It is ideal for deploying continuous integrative servers. However, it is still in its early days and the project requires some risk tolerance to use it in production, which was possible in our case with BOP being a prototype. Kontena was used to deploy Docker, and provides common logging, machine/process statistics, and security. Security is provided at the network or proxy level and is not service specific. Furthermore, Kontena VPN enables the creation of secure networks composed of servers in many locations with local access to all. There is some challenge in using it for big-data.

3.3 Admin Tools

We used the Yahoo Kafka Manager and the Solr Admin UI to manage Kafka and Solr respectively. The Kafka Manager provides support to manage clusters, topic, partitions and replicas. It also allows one to optionally enable Java Management Extension (JMX) polling for broker and topic level metrics and to filter out consumers. The Solr Admin UI is a web interface that makes it easy to view Solr configuration details, run queries and analyse document fields in order to fine-tune a Solr configuration.

4. FUTURE WORK

The BOP is a prototype platform. Subject to funding, there are a number of areas, which can be improved, which include but are not limited to the following:

1. Generality: making it easy to configure a new dataset other than geo-tweets
2. A UI to take advantage of census reverse geocoding and faceting to enable fast, on-the-fly spatial analytics using census variable on big data.
3. Use of Spark for spatial analytics on selected sets.

ACKNOWLEDGEMENTS

We thank the Sloan Foundation and the Boston Area Research Initiative (BARI) for funding, and Harvard Dataverse and for their collaboration. We also want to give thanks to lead developer David Smiley for his innovative contribution to the project, particularly in the areas of core Lucene and Solr enhancements for spatial and temporal data handling, development of the web-service, and the design of geo-enrichment. We would also like to express our gratitude to Ariel Nunez and his team at Terranodo for their innovative work on the BOP web client.

REFERENCES

Smiley, D., and Kakkar, D., 2017. Harvard ABCD-GIS "Billion Object Platform (BOP)- an Open Source, Real-Time, Billion Object Spatio-Temporal Search Platform", Boston, MA, USA http://www.gis.harvard.edu/events/seminar-series/billion-object-platform-bop-open-source-real-time-billion-object-spatio (11 May 2017).

Smiley, D., 2016. Lucene / Solr Revolution "H-Hypermap: Heatmap Analytics at Scale", Boston, MA, USA https://www.youtube.com/watch?v=nzAH5QEl9hQ (11 May 2017).

Lewis, B., 2016. Massachusetts Open Cloud Workshop "Building an Open Source, Real-Time, Billion Object Spatio-Temporal Search Platform", Boston, MA, USA

APPENDIX

Client UI: http://bop.worldmap.harvard.edu/bop/
BOP API: http://bop.worldmap.harvard.edu/bopws/swagger
GitHub: https://github.com/cga harvard/hhypermap-bop
Docker Hub: https://goo.gl/dq6yao
OpenStack https://www.openstack.org/

Revised May 2017