Center for
Geographic Analysis
Harvard University

Investigating Hadoop for Large Spatiotemporal Processing Tasks

David Strohschein dstrohschein@cga.harvard.edu
Stephen Mcdonald stephenmcdonald@cga.harvard.edu
Benjamin Lewis blewis@cga.harvard.edu
Weihe Wendy Guan wguan@cga.harvard.edu

AAG Annual Meeting
Chicago, IL
4/21/2015

Basic Concepts

- **MapReduce**
  - a programming model for processing and generating large data sets
  - with a parallel, distributed algorithm on a cluster
- **Hadoop** is an open source implementation of MapReduce
  - Uses commodity servers
  - Can scale up from a single server to thousands of machines
  - Strong resiliency by software detecting and handling failures
- Hadoop Distributed File System (**HDFS**)
  - Data in a Hadoop cluster is broken down into smaller pieces (blocks) and distributed throughout the cluster
  - Map and reduce functions are executed on smaller subsets of the larger data sets
  - Provides scalability for big data processing
- The Apache Hadoop **YARN** (Yet Another Resource Negotiator)
  - One of the key features in 2nd-generation Hadoop
  - A cluster management technology

When to Use Hadoop

- Large calculation problems suitable for a “divide and conquer” solution
  - Problem can be broken into parts and run separately
  - Answer to one part does not influence the answer to another part
- Large numbers of processors
  - Accessible
  - Affordable
  - Easy to manage

Case 1: Crawling the web for map services to enhance WorldMap

- WorldMap is a web-based, open source, collaborative mapping platform
  - under development at Harvard CGA since 2010
- The National Endowment for the Humanities Implementation Grant sponsored creation of a comprehensive and sustainable map service registry in WorldMap
  - for discovery, creation and sharing of any work that can be represented spatially

Key Objectives

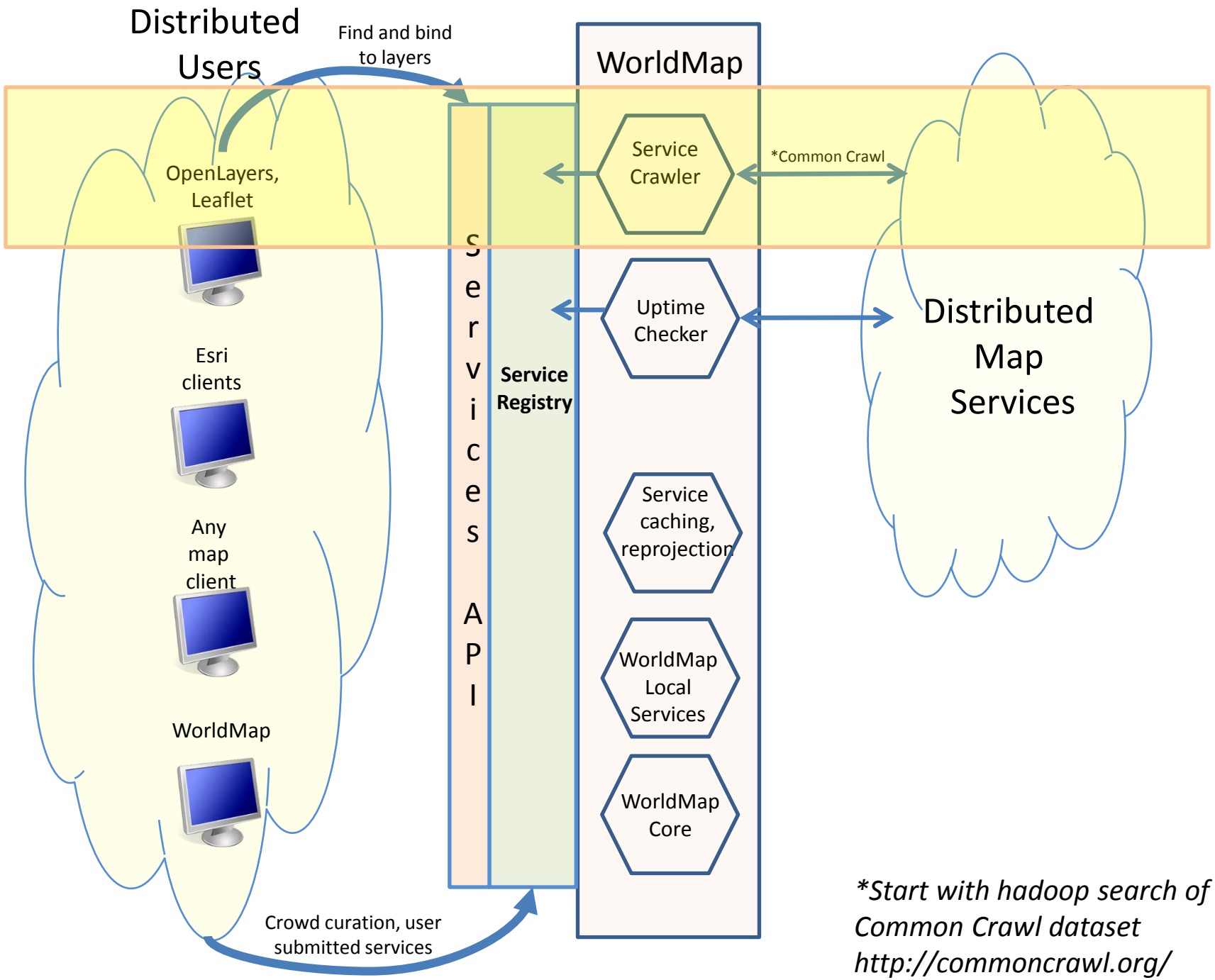
- Uncover the millions of online web map servers and their layers (the dark geoweb)
- Make the layers accessible to anyone within any mapping application
- Take advantage of active and passive crowd curation to improve search in a metadata-weak environment
- Create a public open source data discovery platform anyone can build on and improve

Creating a Global Map Service Registry

- Build registry of web map services (millions of map layers)
- Allow anyone to add new services to the registry
- Maintain uptime statistics on each service
- Provide a fast, faceted search interface
- Use WorldMap usage statistics to improve search
- Make API available so any system can use it
- Eventually bring in stats from systems outside WorldMap which use the API

Building the Registry Open API

- Public, RESTful API
- Access all (public) map layers within WorldMap
- Access all service layers outside WorldMap
- Access all Maps (collections of layers) within WorldMap
- Search on information:
  - Metadata
  - Usage statistics
  - Attribute info (for local layers)



**Start with hadoop search of Common Crawl dataset http://commoncrawl.org/*

Temporal Properties

- Remote map services
  - Last harvest date
  - Uptime statistics
- Data submitted to WorldMap
  - Date data describes (metadata)
  - Date data was created (metadata)
  - Date data was submitted (system tracking)
- Search tool supports search by time

How Many Layers Are Out There?

- We estimate millions, totaling petabytes of data which is currently VERY hard for the average researcher to find and use.
- Try this to estimate number of Esri REST servers (718,000)
  - allinurl: http "arcgis rest services" mapserver -test -kml -kmz -sitemap -query
- Try this to estimate number of WMS servers (30,000)
  - allinurl: http "?request getcapabilities" -test

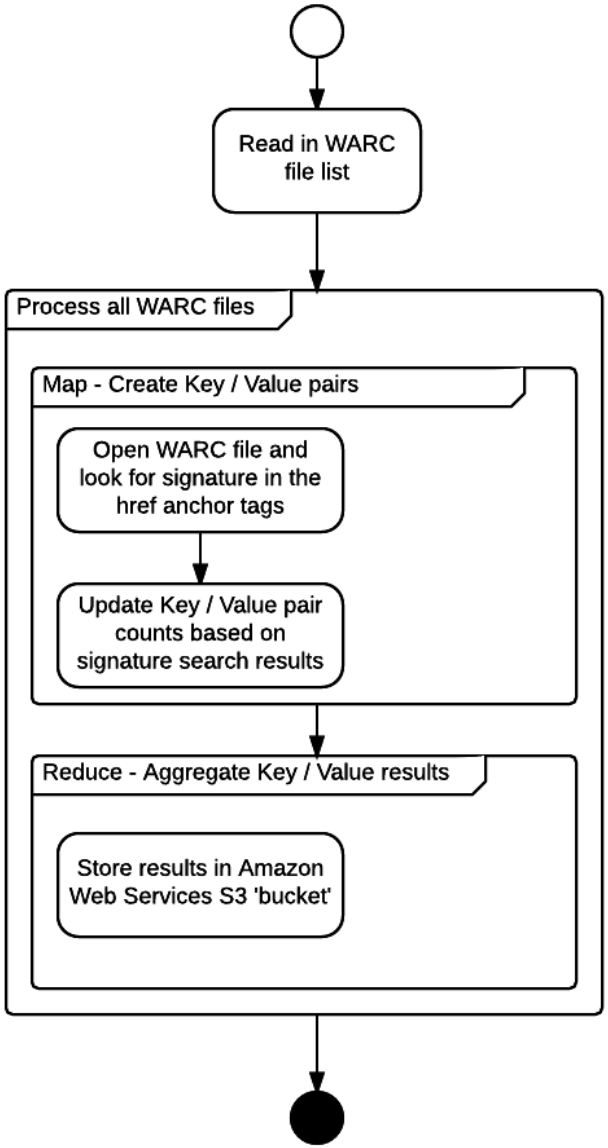
An Approach for Crawling the Web Using Hadoop

- Search web for signatures against the Common Crawl (CC) commoncrawl.org archive
  - Stored as compressed Web Archive (WARC) formatted files on Amazon S3.
  - CC is entire content, less multimedia, of the publicly-available web
- Employ multiple machines to process the data in parallel
  - Avoid investing in hardware by using the Amazon EC2
- Use Hadoop/YARN framework executing Map/Reduce functions to aggregate information about URLs to spatial assets
  - http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html
- Collect URLs for later processing and harvesting

Software Design Considerations

- There is some advantage in using Java since Hadoop is written in Java, but many are using Python and other languages
  - We used Java
- WARC files can be processed with well known Web-Scraping tools
  - We used jsoup http://jsoup.org/
- Some processing is required to prepare crawl output for harvesting
  - We used AWK scripts

Processing a WARC File



- Search WARC files for well-known geospatial service or data signatures.
- A signature is a set of string combinations of interest. The application compares these signatures against the “href” tags within the WARC file being processed.
- For example, those URLs that contain the string "arcgis/rest/services" and the word “mapserver”, but don’t contain the following words: “test”, “kml”, “kmz”, “sitemap”, or “query”, are one type of URL ‘signature’ of interest.

Some signatures We Looked for

- OGC Services
  - Look for "?request getcapabilities" and not “test” in the href URL
- ESRI Rest Services (in the Target-DOMAIN-URI string within the WARC Response Header text)
  - Look for “/arcgis/rest/services” in the target-DOMAIN-URI
- KML or KMZ files
  - Look for an href URL ending in .kml or .kmz
- Compressed shapefiles
  - Look for “shape” or “shp” and string ending with “.zip” in the href URL
- Tile Servers
  - Look for “tile” or “tiles” and string ending with “.png” in the href URL

Sample text from a WARC File

WARC/1.0
WARC-Type: response
WARC-Target-URI: http://vcgi.vermont.gov/warehouse/web_services#maps
WARC-Date: 2014-11-18T13:32:21Z
WARC-Record-ID: <urn:uuid:c44719a1-93bb-de00-4730-87165d2f7d79>
Content-Type: application/http; msgtype=response
Content-Length: 42025

WARC Header Info

•
•
•

Example Signatures

Vermont Parcel Boundaries, VT State Plane Meters:

ArcGIS 10.x Layer File
|
<a href="http://maps.vcgi.org/arcgis/services/EGC_services/MAP_VCGI_VTPARCELS_SP_NOCACHE_v1/MapServer/WMSServer"
target="_blank">
ArcGIS 9.3 or lower and Open Source GIS
Vermont Parcel Boundaries, Web Mercator:

ArcGIS 10.x Layer File

Java Code Signature Search

- Signature search facilitated by the use of several functions designed to detect a class of signatures.
- The example on the right is one such function designed to detect those [geoservice] URLs that are associated with the ESRI REST services.
- When processing a WARC, the HTML text of the response is parsed by jsoup, a Java library designed to separate the contents of an HTML document into a branched structure that will facilitate further processing.
- During this operation, those 'href' strings within an anchor tag '<a> … ' of the document are compared with each signature function for a match.
- If one is found, the signature key receives a 1 for its value.
- The key/value pairs are aggregated later, during the 'reduce' phase of the operation.

ESRI REST Services Search Function

```
//Look for ESRI Rest services, i.e. allinurl: http "arcgis rest services" mapserver -test -kml -kmz -sitemap -query
public static Boolean isArcRestServices(String lowerURL){
   Boolean returnVal = false;
   if (lowerURL.contains("/arcgis/rest/services/") == true && lowerURL.contains("mapserver")== true){
      if (lowerURL.contains("test") == false){
         if (lowerURL.contains("kml") == false){
            if (lowerURL.contains("kmz") == false){
               if (lowerURL.contains("sitemap") == false){
                  if (lowerURL.contains("query") == false){
                     returnVal = true;
                  } else {
                     returnVal = false;
                  }
               }
            }
         }
      }
   }
   return returnVal;
}
```

Lessons Learned

- Learning curve
  - Takes time to understand and implement the Map/Reduce paradigm in the AWS Hadoop/YARN framework
  - But there are multiple sources of useful examples and libraries
- Easy to run out of memory
  - It is easy to get Java heap errors, so configure machines with enough memory to execute
  - But keep memory small enough to make efficient use of instance RAM
- Better performance does not generally cost more
  - Depending on the type and number of instance, the cost is the same – it's the processing time that's different
  - Balance the choice of instance type, memory allocation, and number of instances in order to process the files in an efficient and cost effective manner.

WARC Processing Statistics

Date	WARCs Processed	Cluster Size	Instance Type	Processing Time	Items Found	cost
Dec. 1, 2014	15	1 master 3 slaves	m1.medium 3GB heap size	34 minutes		$0.70
Jan. 20, 2015	13212	1 master 25 slaves	r3.xlarge 4.7GB heap size	20 hours	406016	$364.00
Jan. 22, 2015	13212	1 master 50 slaves	r3.xlarge 4.7GB heap size	10 hours	297569	$357.00
Jan. 23, 2015	13212	1 master 100 slave	R3.xlarge 4.7GB heap size	5 hours	306120	$353.00

Initial Output of Crawl Processes

Date	WARCs Processed	ESRI Servers	OGC Servers	Tile Servers	Shape Files	KML Files
Jan. 20, 2015	13212	521	365	6421	24689	374020
Jan. 22, 2015	13212	667	308	5930	23667	266977
Jan. 23, 2015	13212	446	331	6068	25583	273802

Crowd Curation of Map Services

- Passive
  - Count frequency of URLs from crawl
  - Capture page rank for URLs after crawl
- Active
  - User adds a layer to a map in WorldMap
  - User certifies a layer in WorldMap
  - User ranks a layer in WorldMap

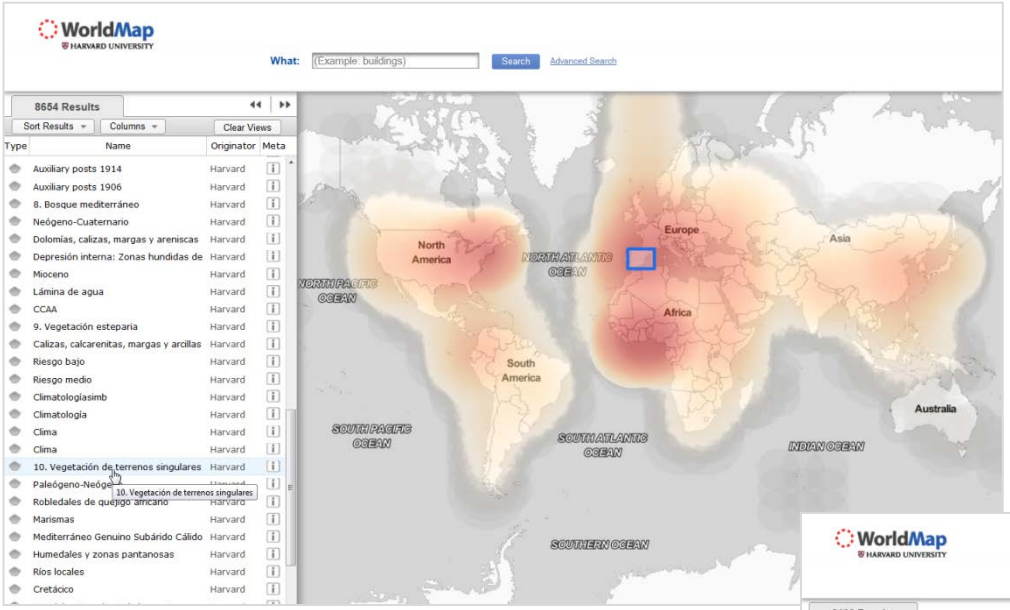
Calculating URL Frequency of Occurrence

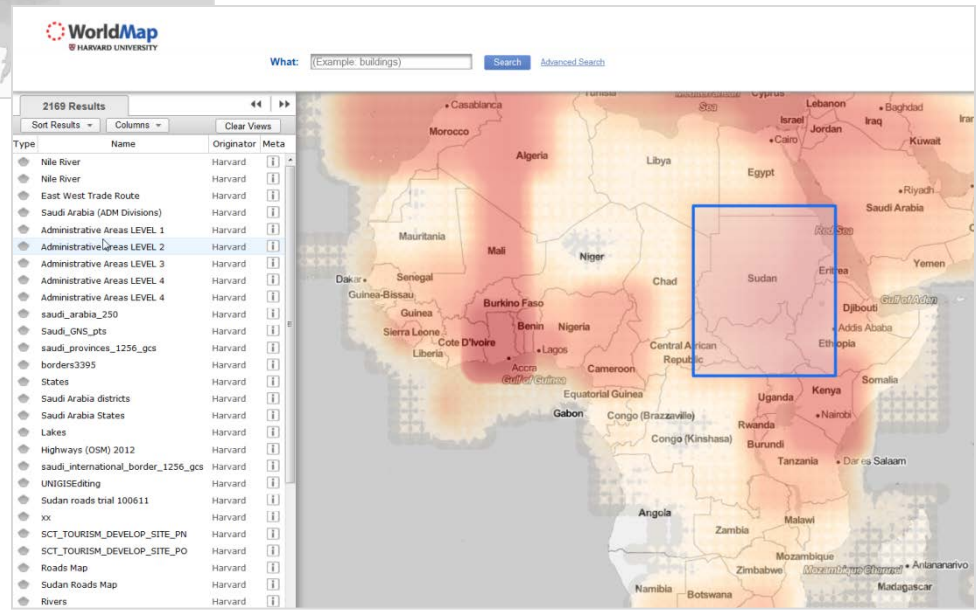
- For every WARC file that is processed, 0-N Key / Value (K/V) pairs are generated.
- Each K/V represents a URL string and the number of occurrences of that URL in the WARC file.
- These K/V pairs are aggregated, by URL, at the end of all WARC file processing.
- A URL's frequency of occurrence is calculated by taking the total number of times it occurred in all the WARC files processed, divided by the total number of all URLs that matched one of the signatures.

Ranking Search Results

- Using Lucene/Solr one can incorporate many factors to rank search results, including:
  - Multiple occurrences of URL on web
  - Page rank of page where URL was discovered
  - Service used in a map in WorldMap
  - Results match key word or synonym
  - Results are toward center of spatial extent defined

Current (ALPHA) Map Service Layer Search Client





Here is an initial version of a search client using Lucene heatmaps against a Solr registry containing overlapping layer footprints for local and remote layers.

Case 2: Query, analyze, and subset global geo-tweets

- We need a Big Spatiotemporal Data Visualization Platform
  - One of our evaluated approached was to use in-memory Hadoop
  - At the end we decided not to use Hadoop for performance reasons
- We developed a solution that is simpler, using Lucene
  - We built spatial heat-mapping into Lucene (with sharding)
  - This solution can scale to visualizations against millions, even billions of features.
  - https://lucene.apache.org/
- We chose the OpenGeoPortal search client and Solr schema
  - for deploying the heatmap search
  - https://github.com/OpenGeoportal/OGP

Case 3: Calculate network distances between thousands of points (billions of calculations)

We plan to parallelize on Hadoop to break job into many pieces, using one of two approaches:

1. Create custom Amazon Machine Instance (AMI) using PostgreSQL/PostGIS with pgRouting libraries
   - pgRouting supports numerous shortest path algorithms
   - http://pgrouting.org/; http://postgis.net/
2. Implement Hive database
   - Hive is a native Hadoop database application with geospatial extensions
   - This requires implementing a shortest path algorithm in basic geospatial SQL

Both approaches require road vectors stored in Amazon Web Services (AWS) Simple Storage Service (S3) for desired performance

Designed Workflow

(not implemented yet)

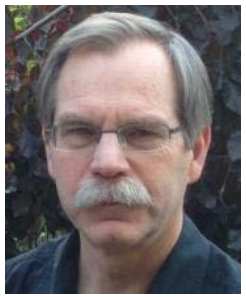
- Use the Hive Geospatial extensions
  - GIS Tools for Hadoop: Big Data Spatial Analytics for the Hadoop Framework
  - http://esri.github.io/gis-tools-for-hadoop/
- Store road vector data files on Amazon S3 storage
- Create a distance node graph from road data
- Use a Java algorithm for Dijkstra's shortest path algorithm
  - http://algs4.cs.princeton.edu/44sp/DijkstraSP.java.html
- Combine this algorithm with the "Spatial Framework for Hadoop" and the "ESRI Geometry API for Java" (from "GIS Tools for Hadoop")
- Execute the code within the Hive framework to leverage off of geospatial calculation capabilities, i.e. ST_DISTANCE( )

Center for
Geographic Analysis
Harvard University

THANK YOU

Investigating Hadoop for Large Spatiotemporal Processing Tasks

David Strohschein dstrohschein@cga.harvard.edu
Stephen Mcdonald stephenmcdonald@cga.harvard.edu
Benjamin Lewis blewis@cga.harvard.edu
Weihe Wendy Guan wguan@cga.harvard.edu



AAG Annual Meeting
Chicago, IL
4/21/2015