Plague Tracker Tool

http://plague.mybluemix.net/
Amy Chou, Carita Ou, Chris Rehfeld, Liping Sun (team 31)

Abstract— Contagious diseases may spread from person to person through physical contact or travel through the air. A common contagious disease such as the flu can be spread through the air before you are sick. Ebola on the other hand, is spread through close and physical contact and is contagious when there are symptoms. We have developed a tool to track who an infected person came in contact with. This paper documents our web solution for limiting the spread of contagious diseases and discusses how we utilize large datasets of human locations to discover other users that might have become infected.

I. Introduction

The world has historically been subject to plagues which destroy mass amounts of human life. Disease spreads rapidly, and due to the exponential nature of contagion, containment is extremely difficult for diseases which spread easily. Generally, the only effective way to contain the outbreak is to isolate the infected people before they can spread it to others. Society doesn't currently have a good way to identify the people who should be isolated, and with the rapid transit systems of modern society, a disaster of unprecedented scale is looming. We propose a system which can help disease control personnel do their job of isolating and containing the plague.

II. Proposed Solution

What would be helpful to disease control personnel would be for them to know who has come in contact or proximity with infected people. This is difficult because people often don't know who they've been in contact with. Luckily, with the increased connectivity in the modern day, tracking data of people's movement patterns is becoming more readily available. The purpose of this project is to gather and utilize these large datasets of people's movement in order to find people who have been in contact with each other during certain periods of time. The result of this project is a user-friendly web application where Disease Control personnel can enter an infected user and related information, and be able to see a visual representation of who that infected user was in contact with. By being able to locate other users who might be infected, the Disease Control center can isolate the disease and prevent it from spreading.

III. User Guide

Going to the Plague Tool, http://plague.mybluemix.net/, health officials have several fields to fill out. These fields help determine who it is the health officials are tracking, starting from when and how much data they want to collect.

A. Original Infected User:

1) **User id** is the identification number representing a user. This number is unique per individual and is used to track users.
2) **Time When Infected** is the time the user is believed to have become contagious with the virus. This is the time the tool will use as a starting point to locate the infected user and map out the other users who became infected as a result.

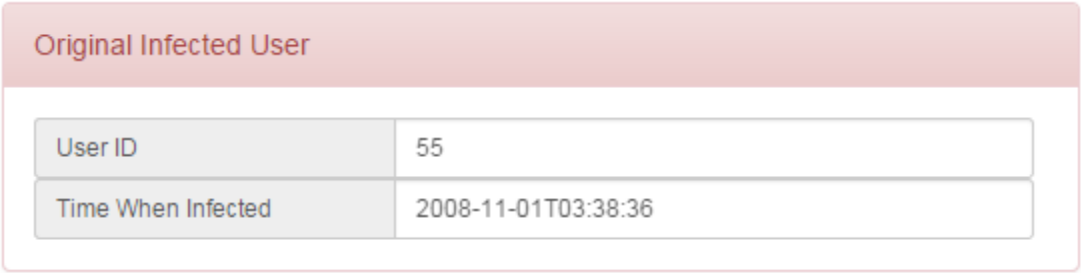

Fig. 1. Original Infected User Form

B. Disease Attributes:

1) **Max Infection hops** is the number of times a disease can hop to another user before it loses its efficacy. For example, user A is the infected person and comes in contact with user B (1st hop). User B then comes in contact with user C (2nd hop) and user C comes in contact with user D (3rd hop). The idea is that some diseases may mutate after hopping too many times, losing their deadliness. Most diseases don't lose their efficacy though, and so a large value such as 10,000 should generally be entered.
2) **Infection proximity** is the distance that 2 users need to get within each other in order for an infection to spread. This number varies by disease as some spread within close contact, while others have larger ranges due to being airborne.
3) **Incubation Time** is the minimum amount of time that a user must carry the disease before they can spread it to other users. For example, some disease are not contagious until symptoms are shown, which may take hours or days. However, other diseases are instantly contagious, and so you could enter a value such as 0 minutes to simulate this.

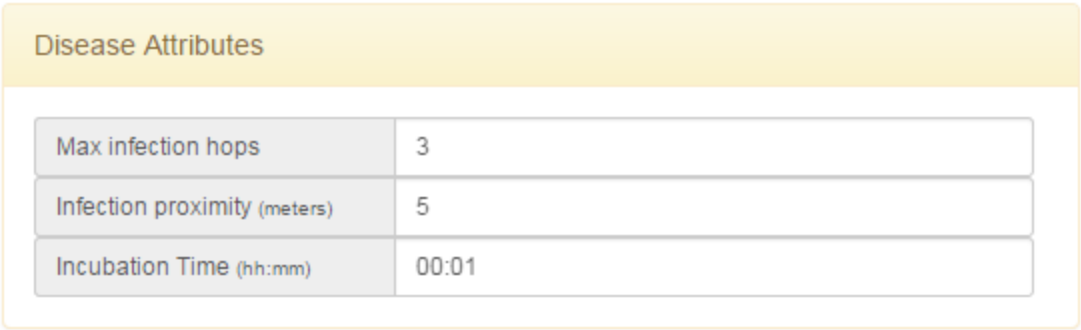

Fig. 2. Disease Attributes Form

C. Analysis Attributes:

1) **Limit Max Infected Users** is an upper bound on how many users the simulation will infect. Because diseases can spread exponentially, there's a possibility that the resulting map of infected users could be saturated with too many data points, making it unreadable. This parameter simply stops the simulation once this many users become infected.
2) **Simulation Duration** is how long the simulation will run for. For example, if you entered a start time of 10am, Dec 1st for the originally infected user, and then entered 7 hours for the simulation duration, then when the simulation finishes running, the map presented to you depicts which users are infected with the disease, at 5pm Dec 1st (7 hours later).

Fig. 3. Analysis Attributes

D. Infection Results

After you submit the form and the app finishes computing the results, it presents a map, with a marker at each location where an infection spread to a new user. It also draws lines between users, which lets you see who infected whom. You can also click a marker, to see the userid of whom that marker represents.

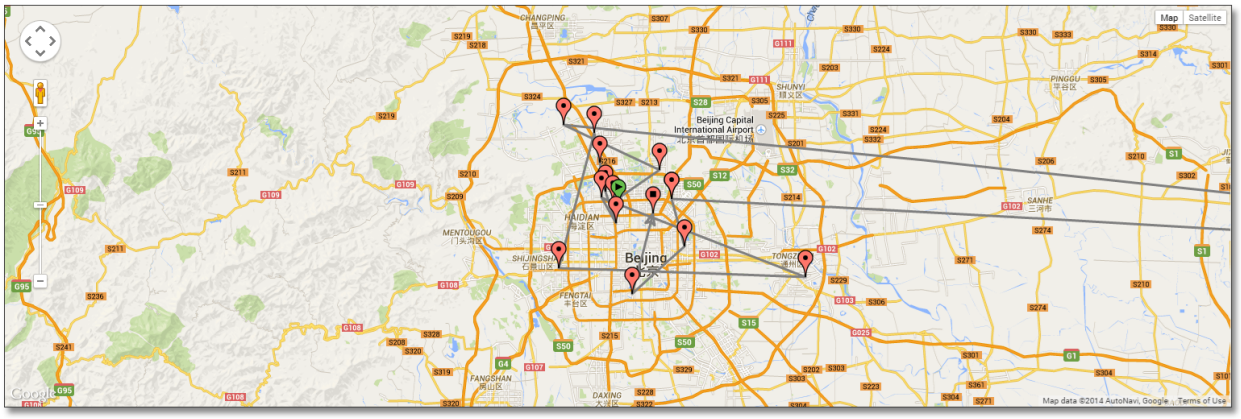

Fig. 4. Infection Results

E. **Quick Start for Demo purposes**

If you enter userid 55, 66, or 888, the rest of the form fields will be ignored, and precomputed data will be plotted on the map instantly. If you want to see the app compute results in real time, you'll need to enter a real userid and start time.

Some suggested values:

userid: 1
time: 2008-11-01T00:42:35
analysis duration: 00:10

The rest of the form values have sensible defaults. Again, be aware the tool is slow, and the above suggested values may take 30 or minutes to compute a result.

You can also use the *Internal Data Exploration Tool* to find other valid userid + timestamps combos to use.

IV. Internal Data Exploration Tool

One of the difficulties of using this tool is entering a valid userid and timestamp from which to start a simulated infection. You need to enter this data to kick off a simulation, but you don't know which data points actually exist in the database, so we created a tool to help you find these data points. At /plague/plot.html, we created a form that takes in two values: date and seconds. This tool will search the database and find records near the entered timestamp. It will produce a maximum of one record per userid, and will present that data on a map with clickable markers. Clicking a marker will show you the userid and the timestamp, which you can then copy into the main tool, using it as data for the originally infected user. This tool helped us find good data that we could use to demonstrate our application. It was especially helpful because it plots the points on a map, so that we could pick data points near other users. In order for an infection to spread, we need to infect a user who tends to be geographically and temporally nearby other users, and the map facilitates finding these prime starting points.

This tool would not be used by real users in a real production environment. It's only for data exploration, to help find data points that would yield interesting results for presentation purposes.

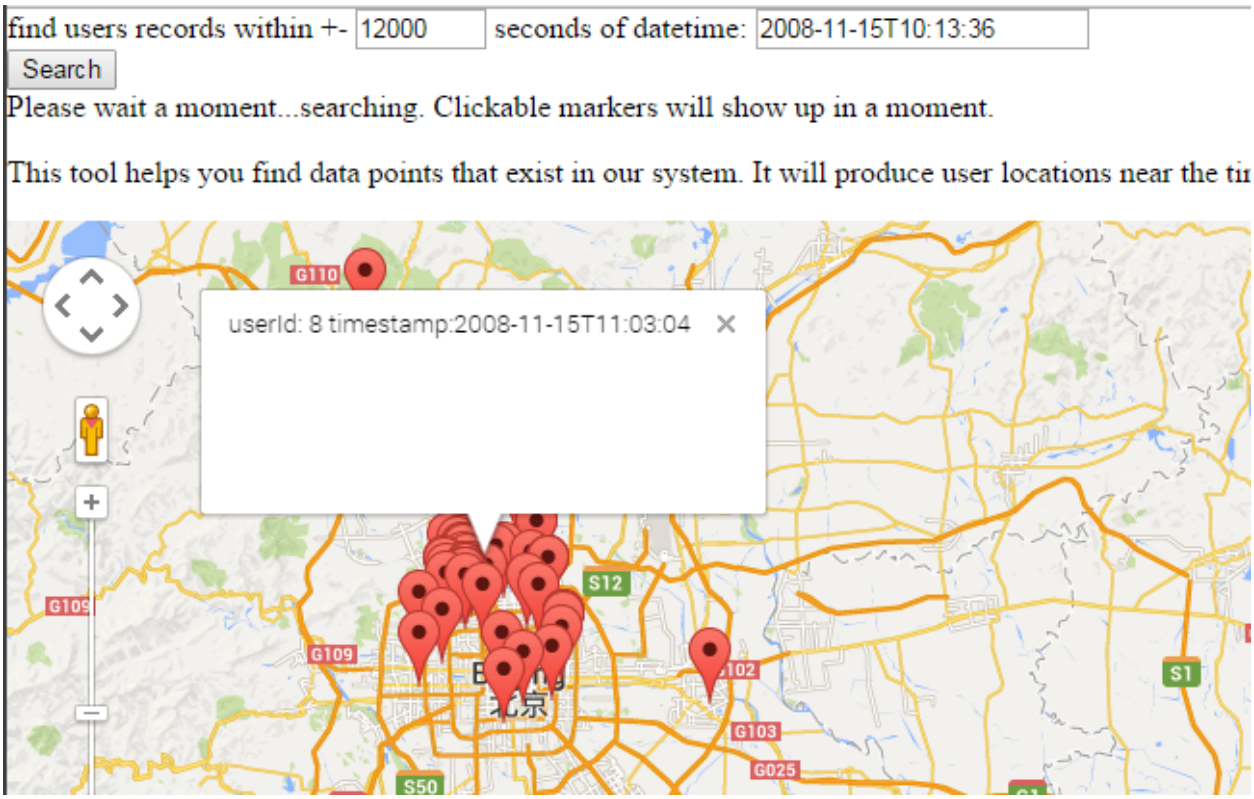

Fig. 5. Internal Data Exploration Tool

V. Functional Requirements

1) The website should have a form that accepts user input about the infected user, and the infection.

2) The server should process the user input and return a list of infected users from the dataset.
3) The infected users and their location of infection should be mapped on the Google Maps.

VI. Nonfunctional Requirements

1) The user interface should be intuitive and easy to navigate.
2) The application should not require software installation in order to be used by end users.
3) The application should display data in a visual manner which enhances data comprehension.

VII. Use Cases

Actor: Center for Disease Control personnel
Precondition: Have the information of the infected user
Flow of Events:

1) A user is identified as being infected by a hospital
2) The hospital informs the CDC, providing the name of the infected user
3) The CDC fills out the form with information of the infected user
4) Click the submit button
5) Allow time for the job to process and for the data points to be loaded onto the map
6) The CDC personnel analyzes the map
7) The CDC personnel clicks user points for more information
8) The CDC personnel takes potential action based on the information

Exceptions:

1) No response
    a) Resend request

VIII. Design Considerations

For the purposes of this project, we expected the size of the data to be quite large. Furthermore, as this project moves forward, we will continue to add more and more data. The more data we can aggregate, the better our users will be able to track and prevent the spreading of these diseases. In order to be able to analyze and process such large amounts of data, we needed to utilize Big Data solutions. In our design considerations, we looked for solutions that would be scalable, powerful, and fast, and that would allow for rapid development given the short timeline. We chose to go with Hadoop to allow for distributed processing on a large data set. In order to help us develop faster given the short timeline, we chose to go with the IBM Analytics for Hadoop. See Section IX: Implementation for more information on this solution.

IX. Implementation

A. Overview

Our back end consists of a Hive data warehouse running in IBM BigSQL on the Bluemix cloud platform. One layer above that is our Java HTTP Servlets. For the front end, we are using the Twitter Bootstrap framework, including HTML5, CSS3, and jQuery along with the Google Maps mapping Api. Upon sending a GET request to the Servlet, the Servlet runs our algorithm on our data set by using BigSQL queries, and eventually returns a JSON result set. We then utilize the Google Maps API in order to plot this result set graphically.

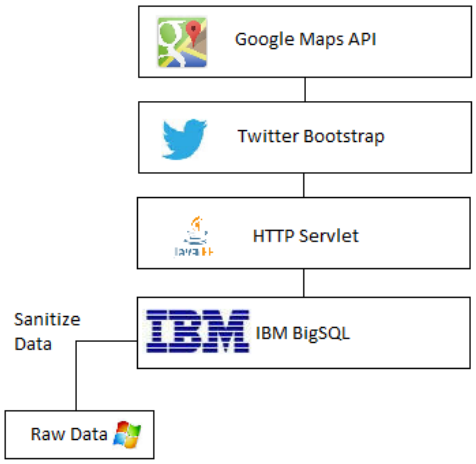

Fig. 6. Architecture Overview

B. Data Layer

Data: The data used for this project was based on a dataset published by Microsoft. This dataset, called GeoLife GPS Trajectories, tracked 182 users in China for around 3 years. These users installed a smartphone app which constantly recorded their location, user id, and timestamp, about every 5 seconds. The users were primarily centered around Beijing. For the project prototype, we decided to limit the amount of data to one month of tracking time. In order to prepare the data to be loaded into the BigSQL database, we wrote a piece of code to clean up the data. This piece of code assigns User ID's to each user and appends them to each row. It also reformats the time into a timestamp that is compatible with the database. Upon analyzing the set, we found that the data is slightly sporadic due to users turning their devices on and off during the time period. Due to the limited amount of users and their trajectories, we used some of the existing data in order to generate some mock data that follows the real life patterns of moving human beings. The reason for generating more mock data is to increase the size and quality of the dataset for purposes of demonstration and proof of concept.

Data Warehouse: The sanitized dataset was loaded into an IBM BigSQL database. We created an "IBM Analytics for Hadoop" application on the BlueMix cloud platform. Then, using IBM BigInsights, we loaded the data into HDFS, and created a Hive table. Using Hive, we were able to quickly write queries and fetch data, benefitting from the power of Hadoop without actually having to write native MapReduce jobs. IBM BigSQL also supports HBase tables, but after considering our needs for this project, we chose to go with Hive. The nature of Hive suits this project better, since it only involves reading and appending data. Using IBM BigSQL, we were able to easily connect our application using to fetch information from the tables using JDBC, although we used the Apache Commons DBUtils package to ease data access.

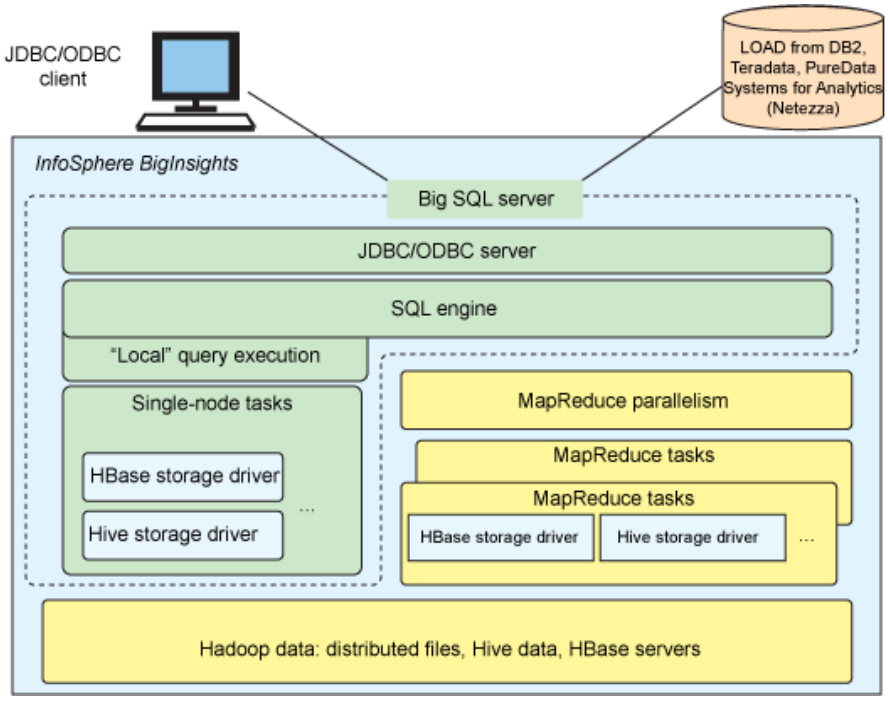

Fig. 7. IBM BigSQL Architecture (Bruce, et. Al.)

C. Web Application

At the application level, we utilized the HTTPServlet from the javax.servlet API. We then implemented the doGet method in the servlet to read information from an http GET request. Upon reading the information from the request, the servlet executes our algorithm to produce a result set. The result is formatted as JSON and returned to the client, where post processing and visualization happens.

D. Front End

For the user interface, we used the Twitter Bootstrap framework with HTML5, CSS3 and jQuery. We chose this framework to enable rapid front-end development that results in a sleek, interactive, well supported and modern website. The use of jQuery and the Twitter Bootstrap framework enabled us to easily implement powerful dynamic features on our website that can be integrated with the rest of our stack. We created a menu bar, as well as a form that can take user input, retrieve information from the HTTP Servlet, and plot it on the Google Maps API. Custom CSS3 stylesheets make the look of the website more visually appealing to the user.

E. Google Maps API

In order to visualize the spread of the disease, we used Google Maps JavaScript API v3. In googlemap.js, we first load the Google Map Canvas into our web page. By using AJAX, we post a query to the server and receive the infection data in JSON format for plotting. Last, we plot the infection path and infected people with user information by using Maker, Polyline, and InfoWindow features in Google Maps API.

F. Deployment

We deployed this application on BlueMix cloud platform so that our application is easily accessible from anywhere. We added the "IBM Analytics for Hadoop" service to facilitate storing and querying the data. Using Eclipse IDE, we created a WAR file from our code, and uploaded it to BlueMix via the cf command line tool.

G. Algorithm

The algorithm which computes the infection graph is an iterative one, which basically loops, computing new infections at progressive 5 second intervals. It starts with the original infected user, and notes the current time. It looks for other users who are nearby that user at that instant in time. If conditions are right, these nearby users become infected. Then, the time is incremented by 5 seconds, and the process is repeated, except this time, it searches for users nearby the locations of **any** of the currently infected users. This process continues until the max simulation time parameter is reached. Obviously, the real algorithm we use is much more complex because it needs to respect all the input parameters that our app accepts, but this explanation just aims to describe the high level approach we use.

X. Limitations

This app cannot tell you all the users who truly get infected. One of the biggest limitations is that we will never have data for every user on the planet, and so there will be users who get infected who weren't shown in the simulation. Additionally, just because our app says a user became infected, obviously doesn't meant they really did. But, this is a false positive, and in the context of the problem were trying to solve, false positives are acceptable.

XI. Future Scope

In order to make this application be effective in the real world we would need to crowdsource more data, sanitize it as it comes in, and stream it into our database so that we have a semi real time data set. Being able to get data set which includes people around the world would be one of the challenges we would certainly face whilst extending the scope of this project. However, we assert that these data sets exist today, and are becoming more and more popular every day. The Google Maps app uses crowd sourced data to power its traffic congestion feature, and this is installed by millions of users. It coincidentally collects the necessary data to power our app.

Our application could use some UI improvements. Given the short timeline of the project, we have not considered usability extensively. At the moment, the user must refresh the page every time they wish to submit another query. It also currently does a poor job of reporting errors; it should guide the user towards resolution.

We may be able to make some code performance optimizations in the future. Currently, each time the user submits a query, it is taking about one minute to walk through around one minute of simulation data. For example, this means it takes an entire day to simulate the infections which occurred on some arbitrary day. This can certainly be improved for better performance and user experience by using more suitable data structures, and in particular, spatial indexes.

XII. Bluemix Feedback

We had several problems using Bluemix. While using the "IBM Analytics for Hadoop" application, we did not see it mention anywhere on the page that the service would soon expire. We also got no notifications that the service would expire before it actually did expire. It came as a surprise to us when the service expired, and it did not even give an option to pay. We had to reupload all the data on another team member's Bluemix account. This experience was quite bad and we feel that it could be much improved.

We liked the web GUI for Bluemix though. And once we got it set up, it seemed to "just work", which was invaluable and time saving. Deploying our war file turned out to be very easy. What wasn't easy, was finding out how

to do it. We feel we did reasonable web searches for popular terms that developers would use when wanting to deploy a war file, but answers weren't obvious. As such, we feel it's a shame that the documentation is a bit lacking, because the technical capability is there, and implemented very well.

XIII. Conclusion

This project has a lot of potential to help many people. After analyzing the data, we found that many people did in fact come in contact with others, especially in crowded areas such as downtown Beijing and public transit areas. If we can get more extensive data in the future and improve the processing power of our application, we can definitely scale this application. This application can potentially help million of users by preventing the spread of disease and giving people peace of mind.

References:

[1] IBM. (2013, June 14). *What's the big deal about Big SQL?* [Online] Available: http://www.ibm.com/developerworks/library/bd-bigsql/

[2] Microsoft. (2012, August 9). *GeoLife GPS Trajectories*. [Online] Available: http://research.microsoft.com/en-us/downloads/b16d359d-d164-469e-9fd4-daa38f2b2e13/