9/24/2014

Creating More Accurate Stop Times

This document provides a summary of the work completed on the World Bank project for using historic Automatic Vehicle Location (AVL) information, obtained via GPS, to create more accurate schedule times. This document is to summarize the work complete by the end of the World Bank fiscal year on June 17th and also to provide instructions on how to use the software.

# Overview

The purpose of the software is to use GPS based AVL data to create a more accurate schedule information for a GTFS stop\_times.txt file. The value of creating the more accurate schedule information is that it will be used for metrics to better understand the workings of the transit agency and it will also enable trip planners to use more accurate times for planning trips for passengers. The greater accuracy will make the trip plans much more useful.

There are three steps required for generating the more accurate GTFS stop\_times files:

1. Loading existing GTFS configuration data into the database
2. Processing GPS based AVL data to determine accurate arrival and departure times for each stop.
3. Using statistics to determine optimal schedule based on arrival and departure times and output that information into new GTFS stop\_times.txt files.

Data is handled on a per trip basis. This means that for regular schedule based systems that run a trip only once per day, you need several days of data to get multiple data points. But for frequency based configurations where a trip is run multiple times a day, such as for Zhengzhou, each run during the day for a particular trip is processed into a single value.

This software also generates schedule adherence information so one can determine what the schedule adherence was with the old stop\_times and what it would be with the new ones. This allows one to see directly what kind of improvement using the new stop\_times will provide. The schedule adherence information is included in the log file, not in the new stop\_times files.

The results are output into the original GTFS directory as two new stop times files: stop\_times.txt\_new and stop\_times.txt\_extended. Note that the original stop\_times.txt file will not be overwritten. If you want to use the new stop\_times.txt\_new file you need to change its name to stop\_times.txt, thereby overwriting the old file.

The stop\_times.txt\_new file has exactly the same format as the standard stop\_times.txt file. The stop\_times.txt\_extended file contains the standard data but also adds some very useful columns including the original stop time so you can see how much it is being changed, the min and max arrival & departure times, the standard deviation so you can see the distribution of times, and the number of data points so you can see how many trips were used to generate the values.

The order of the rows in the new stop\_times files will not necessarily be the same as the original stop\_times file. If the ordering of the original stop\_times file is adequate such that trips are grouped together and that the stop\_sequence increases within the trip, then the new stop\_times files can have the same order. But if there are issues with the ordering of the data in the original stop\_times file, which is somewhat common, then the data is first sorted so that can determine the first stops of trips, which is important for the GTFS data is frequency based. This leads to a different ordering for the stop\_times.txt\_new and stop\_times.txt\_extended files.

To process the data this class reads in arrival and departure data from the database. It batch reads the data 500,000 datapoints at a time, a value chosen to make db reading quick (want a high number) without using too much heap memory at once (want a low number). The arrivals and departures data is read into a map. The map is keyed on routeId so that can handle each route separately (though this isn't truly needed). The data is simply stored as Integers indicating the time of day of the arrival or departure. When reading in departures it also puts the trip departure times into a departureTimesFromTerminalMap so that can determine elapsed time for when frequency based trips are used.

Once all of the arrival and departure times have been processed into a map statistics is used to determine which is the best arrival/departure for the stop\_times output. The goal is to use a time such that only the a desired fraction of arrivals/departures will be early. For example, if you want only 20% of the vehicle to be early with respect to the schedule time, which is reasonable because for passengers it is better for vehicles to be late rather than early so they don't miss the vehicle, then the value should be 0.2. This desiredFractionEarly value is specified when the ScheduleStatistics object is constructed.

The way the software tries to achieve the desiredFractionEarly is by assuming there is a Gaussian distribution of the times. By using the standard deviation of a Gaussian distribution the software estimates the value to use to such that desiredFractionEarly will be attained. Of course the distribution is not truly Gaussian. Therefore several iterations are used to adjust the value in order to get the desired results.

The results are then output into the stop\_times.txt\_new and stop\_times.txt\_extended files described above.

# Status

The software has been created to load the original GTFS configuration information, process the GPS based AVL data into accurate arrival and departure times, and output the more optimal GTFS stop\_times files. The data has been run for Zhengzhou for a full day of GPS data from 4/1/2014. The GTFS configuration used is version “test102”. The amount of data processed successfully is very large and includes approximately 8 million GPS reports resulting in 1.7million arrivals or departures. The resulting GTFS stop times files stop\_times.txt\_new and stop\_times.txt\_extended are included as part of this package so that you can easily see the results of the software without having to run it yourself.

The data in the stop\_times.txt\_extended file is especially useful because it shows details on the data including the original schedule times, min times, max times, standard deviation, number of data points, etc. By examining the data one can see that the system is providing the desired output.

# Additional Documentation

The software is available for browsing on GitHub and can be seen at <http://transitime.github.io/core/> . The “View on GitHub” on that page will enable you to peruse the software on the web without downloading it. Each software class and package is documented in “Javadoc” format and it can be seen on the same GitHub page or by going directly to <http://transitime.github.io/core/javadocs/> .

# General Software Setup

The Transitime software for this project is very extensive and takes a good deal of effort to setup. It therefore takes some work to setup the system to process the data. It is key that you first follow the instructions for installing Java, setting up the database, configuring logging, and running the software. These steps are outlines below.

Note: it has been found that due to a very large amount of data being processed, processing the AVL data into arrivals/departures requires at least 2GB or RAM just for the single process. The computer that does the processing must therefore have at least 4GB of RAM.

## Installing Java

Make sure you have the Java Developers Kit (JDK) available. You need the JDK as opposed to the Java Runtime Environment (JRE) because only the JDK includes the java compiler for compiling the provided software. You also need to make sure that the JDK directory is specified in your PATH environment variable. To determine if you have the JDK installed and accessible via your

PATH, in a command line shell window enter the command “javac –version”. You should have

at least version 1.6 installed. If you do not already have the JDK installed you can download it

from http://www.oracle.com/technetwork/java/javase/downloads/index.html . If the JDK is

already installed but not available on your PATH you will need to add it to your PATH. For Linux

you can use a command such as “PATH=$PATH:/java/jdk1.7.0\_25/bin”. If you are using Microsoft

Windows you can follow the instructions at a site such as http://www.java.com/en/download/

help/path.xml .

## Setting up the Database

A key issue is in setting up a database. The documentation for setting up such a database is far beyond the scope of this document and is best left for someone who has expertise with doing such.

It is recommended that you use MySQL database because it is open-source, widely available, the drivers are included in the transitime.jar,and was used for the testing of the software. It is recommended that you work with a IT specialist who has setup such a database because it can be somewhat difficult. You will need to setup the tables in the database using the SQL script ddl\_mysql\_org\_transitime\_db\_structs.sql that is included in with this document. If you are instead using a Postgres or Oracle database you will need to use ddl\_postgres\_org\_transitime\_db\_structs.sql or ddl\_oracle\_org\_transitime\_db\_structs.sql . The name of the database should be the same as the name of the agency, such as “zhengzhou”.

For MySQL, you can get the MySQL Installer and MySQL Workbench from <http://dev.mysql.com/downloads/windows/> . Use MySQL Workbench as the user interface to db. Connect to the db and then click on the Create New Schema icon (looks like a yellow oil drum) to create an instance of the database for a particular transit agency. Then use the file icon to load in the ddl\_mysql\_org\_transitime\_db\_structs.sql schema. Then select everything after the “alter table” sql commands (those will not work if the tables haven’t been created yet) and then click on the yellow lightning bolt icon to execute the SQL. This will setup all the tables so that data can be written to them.

## Database Configuration

If you will be using MySQL as the database then the existing DIRECTORY/hibernate.cfg.xml file can be used out without any modifications. If you are using Postgres or Oracle then you will need to specify the appropriate hibernate.dialect parameter.

Currently the database host, user name (login), and password are configured via command line options instead of in the hibernate.cfg file. This way multiple projects can use the same hibernate.cfg file. In the future it might be decided to simply use the configuration data in hibernate.cfg.file since that is a more standard way of configuring those parameters. The command line options are:

* -Dtransitime.db.dbName=zhengzhou (name of the database)
* -Dtransitime.db.dbHost=localhost (or can be name of a remote machine)
* -Dtransitime.db.dbUserName=root (login)
* -Dtransitime.db.dbPassword=transitime (password)
* -Dtransitime.db.dbType=mysql (mysql or postgres. mysql is the default)
* -Dtransitime.hibernate.configFile=C:/<DIRECTORY>/hibernate.cfg.xml (you will need to set this to use the proper directory)

## Logging Configuration Files

The logback.xml files specify what data should be logged into which log files. This is a very powerful feature but can take a while to learn. There are several config files that are used for different applications. They are logbackGtfs.xml for when processing the GTFS data into the database, logback.xml for processing the AVL data into arrival and departure times, and logbackUpdateGtfsSchedule.xml for when updating the GTFS stop\_times file.

There is only parameter you need to change: name="LOG\_FILE\_ROOT". Change the value to point instead of to “D:/Logs/” to an appropriate directory on the machine that will be processing the data. Simply using “/Logs/” is likely to be adequate if you have permission to create the /Logs directory.

The logback feature is quite powerful but complicated. Full documentation can be found at <http://logback.qos.ch/manual/> .

# 

# 

# Running the Software

Note: the commands are long but need to be entered as a single command. You will need to replace the DIRECTORY, LOGGING\_DIRECTORY, GTFS\_DIRECTORY, and DATA\_DIR with the directory names where the config files, the log files, the GTFS data, and the GTFS-RT data respectively are located. Also note that commands are configured for zhengzhou. You will need to change the agencyId and dbName to the appropriate agency if not working on Zhengzhou data. And lastly, you need to change PASSWORD to the appropriate value.

It is recommend that you cut and paste into an editor the command below . Then you can use the editor to set all of the parameters. Once you have set all the parameters using the editor you can cut & paste the results into a terminal (command prompt) window to actually run the software.

Note that each line of the command below ends with a “\” character. This is so that all the lines will be executed as a single command when you cut & paste it into a terminal window. But if you are using a Microsoft Windows Command Prompt window you will need to change each occurrence of “\” with “^” for all lines to be combined together into a single command.

## Testing database configuration

As a first step it is best to try running a test program to make sure that you can configure the command line parameters properly and actually access the database. The command for testing is shown below. Once you have the database configured try editing the parameters below to correspond to your system and then run the command in a terminal/command prompt window. A good deal of information will be logged to the terminal window. The program will log whether it was able to access a table in the database or whether there was an error.

The test command for you to edit and try out is:

java -Dtransitime.db.dbName=zhengzhou \

-Dtransitime.db.dbHost=localhost \

-Dtransitime.db.dbUserName=root \

-Dtransitime.db.dbPassword=PASSWORD \

-Dtransitime.db.dbType=mysql \

-Dtransitime.hibernate.configFile=/DIRECTORY/hibernate.cfg.xml \

-cp "/DIRECTORY/transitime.jar" \

org.transitime.applications.DbTest

Parameter descriptions:

* -Dtransitime.db.dbName - Name of the database
* -Dtransitime.db.dbHost - Machine where the database resides
* -Dtransitime.db.dbUserName - Login for the database
* -Dtransitime.db.dbPassword - Password for the database. Make sure you set this to the proper value.
* -Dtransitime.db.dbType - Whether using mysql or postgres
* -Dtransitime.hibernate.configFile - Where to find the hibernate database configuration file.
* -cp - The class path where to find the transitime jar file.

## Loading GTFS Configuration Data

This step processes the GTFS configuration data and loads it into the already setup database.

java -Dtransitime.core.agencyId=zhengzhou \

-Dlogback.configurationFile=/DIRECTORY/logbackGtfs.xml \

-Dtransitime.logging.dir=/LOGGING\_DIRECTORY/ \

-Dtransitime.db.dbName=zhengzhou \

-Dtransitime.db.dbHost=localhost \

-Dtransitime.db.dbUserName=root \

-Dtransitime.db.dbPassword=PASSWORD \

-Dtransitime.db.dbType=mysql \

-Dtransitime.hibernate.configFile=/DIRECTORY/hibernate.cfg.xml \

-Xmx1000M \

-cp "/DIRECTORY/transitime.jar" \

org.transitime.applications.GtfsFileProcessor \

-gtfsDirectoryName /GTFS\_DIRECTORY/zhengzhou/test102 \

-pathOffsetDistance 4.0 \

-maxDistanceForEliminatingVertices 3.0 \

-maxTravelTimeSegmentLength 120.0

Parameter descriptions:

* -Dtransitime.core.agencyId - ID of the project. Specifies directory for logging.
* -Dlogback.configurationFile - Where to find the logback logging configuration file. Make sure you set it to the proper directory.
* -Dtransitime.logging.dir - Where the resulting log files are to go
* -Dtransitime.db.dbName - Name of the database
* -Dtransitime.db.dbHost - Machine where the database resides
* -Dtransitime.db.dbUserName - Login for the database
* -Dtransitime.db.dbPassword - Password for the database. Make sure you set this to the proper value.
* -Dtransitime.db.dbType - Whether using mysql or postgres
* -Dtransitime.hibernate.configFile - Where to find the hibernate database configuration file.
* -Xmx1000M - Specifies to allocate 1000MB to the process.
* -gtfsDirectoryName - Where to find the original GTFS data
* -pathOffsetDistance - Shape information for routes is often street centerline data. In order to have the paths be on the sides of the street one can specify an offset so that the resulting paths will be more accurate.
* -maxDistanceForEliminatingVertices - Sometimes the configuration has an unneeded fineness with respect to the shapes, providing more data points than are needed. This parameter configures
* -cp - The class path where to find the transitime jar file.

## Processing GTFS-realtime Data to Generate Arrivals/Departures

This step processes a set of GTFS-realtime Vehicle Position files and uses the AVL data to match vehicles to their assignments and then generates arrival and departure information that is stored in the database.

java -Dtransitime.core.agencyId=zhengzhou \

-Dtransitime.modules.optionalModulesList=org.transitime.avl.BatchGtfsRealtimeModule \

-Dtransitime.core.storeDataInDatabase=true \

-Dtransitime.core.pauseIfDbQueueFilling=true \

-Dtransitime.core.onlyNeedArrivalDepartures=true \

-Dtransitime.core.allowableEarlySeconds=2400 \

-Dtransitime.avl.timeoutSecs=480 \

-Dtransitime.arrivalsDepartures.maxStopsWhenNoPreviousMatch=3 \

-Dlogback.configurationFile=/DIRECTORY/logback.xml \

-Dtransitime.logging.dir=/LOGGING\_DIRECTORY/ \

-Dtransitime.avl.gtfsRealtimeFeedURIs="file:///DATA\_DIR/gtfsRealtimeData1;file:///DATA\_DIR/gtfsRealtimeData2;file:///DATA\_DIR/gtfsRealtimeData3;file:///DATA\_DIR/gtfsRealtimeData4;file:///DATA\_DIR/gtfsRealtimeData5" \

-Dtransitime.db.dbName=zhengzhou \

-Dtransitime.db.dbHost=localhost \

-Dtransitime.db.dbUserName=root \

-Dtransitime.db.dbPassword=PASSWORD \

-Dtransitime.db.dbType=mysql \

-Dtransitime.hibernate.configFile=/DIRECTORY/hibernate.cfg.xml \

-Xmx2200M \

-cp "/DIRECTORY/transitime.jar" \

org.transitime.applications.Core

Parameters:

* -Dtransitime.core.agencyId - ID of the project. Specifies name of database to use. Also specifies directory for logging.
* -Dtransitime.modules.optionalModulesList - Semicolon separated list of modules to initiate. Used to initiate appropriate AVL module.
* -Dtransitime.core.storeDataInDatabase - Set to true so that arrivals/departures will actually be stored in database.
* -Dtransitime.core.pauseIfDbQueueFilling - Set to true so that system will pause temporarily if the database queue is filling up.
* -Dtransitime.core.onlyNeedArrivalDepartures - Set to true so that system doesn’t generate and store data that is unneeded for generating schedule information. This includes prediction data and storing AVL data in database. Setting it to true causes system to run significantly faster and use less memory.
* -Dtransitime.core.allowableEarlySeconds - How early a vehicle can be with respect to its schedule. Had to set it to 2400 seconds (40 minutes) because found that the original GTFS schedules could be very off.
* -Dtransitime.avl.timeoutSecs - How long a vehicle can not report for before it is unassigned. Had to use a relatively large value of 8 minutes because found that sometimes, though rarely, would get gaps in GPS data of 6-7 minutes.
* -Dtransitime.arrivalsDepartures.maxStopsWhenNoPreviousMatch
* -Dlogback.configurationFile - Where to find the logback logging configuration file. Make sure you set it to the proper directory.
* -Dtransitime.logging.dir - Where the resulting log files are to go
* -Dtransitime.db.dbName - Name of the database
* -Dtransitime.db.dbHost - Machine where the database resides
* -Dtransitime.db.dbUserName - Login for the database
* -Dtransitime.db.dbPassword - Password for the database. Make sure you set this to the proper value.
* -Dtransitime.db.dbType - Whether using mysql or postgres
* -Dtransitime.hibernate.configFile - Where to find the hibernate database configuration file.
* -Xmx2200M - Specifies that program can use up to 2200MB or memory, which is about 2.2GB. Yes, this part of the process takes a great deal of memory!
* -cp - The class path where to find the transitime jar file.

## 

## 

## Using Arrivals/Departures to Generate new GTFS stop\_times Files

This step reads the arrivals and departures from the database along with the original stop\_times.txt GTFS file and creates two new stop times files called stop\_times.txt\_new and stop\_times.txt\_extended. The “new” file contains the new schedule information in the standard GTFS stop\_times.txt format. The “extended” file contains the new schedule information and also additional information such as minimum, maximum, standard deviation, and number of data points. It is very useful for better understanding the data.

java -Dtransitime.core.agencyId=zhengzhou \

-Dlogback.configurationFile=/DIRECTORY/logbackUpdateGtfsSchedule.xml \

-Dtransitime.logging.dir=/LOGGING\_DIRECTORY/ \

-Dtransitime.db.dbName=zhengzhou \

-Dtransitime.db.dbHost=localhost \

-Dtransitime.db.dbUserName=root \

-Dtransitime.db.dbPassword=PASSWORD \

-Dtransitime.db.dbType=mysql \

-Dtransitime.hibernate.configFile=/DIRECTORY/hibernate.cfg.xml \

-Xmx1500M \

-cp "/DIRECTORY/transitime.jar" \

org.transitime.applications.ScheduleGenerator \

-gtfsDirectoryName /GTFS\_DIRECTORY/zhengzhou/test102 \

-b 2014-4-1 \

-e 2014-4-1 \

-f 0.2 \

-allowableFromMean 480

Parameters:

* -Dtransitime.core.agencyId - ID of the project. Specifies directory for logging.
* -Dlogback.configurationFile - Where to find the logback logging configuration file. Make sure you set it to the proper directory.
* -Dtransitime.logging.dir - Where the resulting log files are to go
* -Dtransitime.db.dbName - Name of the database
* -Dtransitime.db.dbHost - Machine where the database resides
* -Dtransitime.db.dbUserName=root - Login for the database
* -Dtransitime.db.dbPassword - Password for the database. Make sure you set this to the proper value.
* -Dtransitime.db.dbType - Whether using mysql or postgres
* -Dtransitime.hibernate.configFile - Where to find the hibernate database configuration file.
* -Xmx1500M - Specifies that program can use up to 1500MB or memory, which is about 1.5GB. Yes, this part of the process takes a great deal of memory!
* -gtfsDirectoryName - Where to find the original GTFS data and also place the resulting new, improved stop\_times.txt\_new and stop\_times.txt\_extended files.
* -b & -e - the begin and end dates for which to process data. The date must be specified in the format that is configured for the computer running the software. If it is a Chinese configuration then the format is most likely yyyy-mm-dd (such as 2014-9-29) but if it is a United states configuration then the format is most likely mm-dd-yyyy (such as 9-29-2014)
* -f - fraction of arrivals and departures to be earlier than the new schedule time. A value of 0.2 is usually reasonable.
* -allowableFromMean - For filtering out bad data points. If arrivals or departures are further than this value from the mean they are filtered out.

# 

# 

# Work still to be Completed

The software can now successfully generate more accurate GTFS stop times based on AVL data. There are still several tasks that should be undertaken to complete the project. These are:

* Work with the transit agencies to obtain improved versions of the GTFS configuration data and test it with the system. A significant number of issues were found with the Zhengzhou data that provided for testing. The software should be tested with a more final version of the data. In addition, GTFS data has only been available for Zhengzhou but not the other agencies. Will need to test the software with the other agencies and make any necessary modifications in order to ensure that it works.
* Further test system by investigating any problems that can be found in the resulting data. Though a great deal of effort went into making the system work as possible it should be expected that there will still be some refinements that need to be made based on finding data issues caused by specific situations.
* Attempt to work with more than just a days worth of data. Make sure that software can handle multiple service classes, such as weekends versus weekdays, if so configured (Zhengzhou only uses a single service class in its GTFS data).
* Test system and make any necessary refinements by processing data for the other transit agencies that are part of this project.
* Enhance documentation. The current documentation is extensive but still needs significantly more details and instructions. Also, the documentation should be improved upon based on feedback from users at the transit agencies.
* Provide instructions on how to operate system in cloud using Amazon AWS. The software has been successfully run on a simple laptop with only a dual core processor and 4GB RAM. But given the large amount of data being processed it would be better to be able to also run it using AWS cloud where faster servers can be rented. This will prevent running into problems from lack of computer resources.
* Add enhancements based on feedback from users.