Middleware Project Report

Real Time Data Aggregation and Visualization

Group 12
Deepthi Devaiah Devanira - 65848248
Kevin Sebastian - 44322980
Sachin Bangalore Raj - 28568830

June 2017

1 Abstract

The objective of our Middleware project is a create an application for visualization of correlation between weather, traffic and air pollution in various cities.
With the advent of the ongoing public discourse on climate change and its effects, this project aims to capture patterns in weather and air quality in various cities around the world and to also relate different types of traffic incidents as well as traffic density patterns in congested cities.
The scope of this project includes -

- Building a scalable application that is able to process and store data streams in real-time.
- The system must be robust and should be able to cope up with abnormal situations such as high volume of data , low bandwidth etc
- The visualization tool must be flexible and easily usable. It should provide accurate results

2 Related Work

The implementation involves live data streaming, data ingestion, cloud storage and visualisation. Since we have our custom implementation for data ingestion similar to spark map-reduce model, we are taking into consideration issues like dynamic load balancing and load shedding.
The paper by Zhang et al [**1**] establishes that traffic congestion results in increased vehicle emissions and degrades air quality significantly. The paper [**2**] shows an analysis of the relationship between vehicle traffic and air pollutants in Beijing. The paper [**3**] formalizes the load shedding problem by converting it into an optimization problem with an objective function that provides a measure of how much the output would be affected by the absence of discarded data. The paper by Wingerath et al [**4**] on Real Time Stream Processing for Big Data provides an analysis of state of the art stream processors for analysing big data by comparing several popular tools on the market such Storm, Samza and Spark Streaming.
The paper [**5**] has an approach similar to that of [**3**]. They have also defined load shedding as an optimization problem with an objective function. However, they generate a set of static load shedding plans in advance and apply the appropriate plan under different states of the input. The paper proposes both a distributed as well as centralized approach towards solving the problem. The paper assumes that data arrives in bursts and as a result the volume of incoming data can vary significantly. The objective remains the same - minimize degradation in output quality. Load balancing algorithms are used to improve the performance of Distributed Systems. The paper [10] explains about several distributed load balancing techniques, merits, demerits and comparison between different distributed architecture.
In our project architecture, we need to store large volume of streaming data. For obvious reasons, storing them on local machines is not feasible in every sense considering availability and scalability. So, the better

alternative is to store all the data in the cloud. All enterprise storage is moving especially towards public cloud storage namely AWS, Azure. The storage cost on these public cloud services is very cheap which makes them even more attractive. As stated in [**6**] cloud systems not only offer storage as a service but also infrastructure, platform and software as services. It is very essential to build the cloud architecture in a standard manner so that it can incorporate the various needs of different enterprises.
Paper [**7**] explains a very good generic architecture for cloud systems. The paper suggests the need for scalability and I/O efficiency in cloud architecture and also emphasizes on the lack of research in this direction. Another scalable cloud architecture is proposed in [**8**] namely ecStore. It also follows a layered architecture with different layers handing different functionality. It has is a distinct layer for replication and transaction management.
The three main areas of focus of research related to this paper were Load Shedding, Load Balancing and Cloud Storage. From a Distributed Systems and Middleware perspective, these topics are the most important components of our project and their implementation affects greatly the accuracy and efficiency of our results.

3 Architecture

The high-level view of the architecture is shown below.

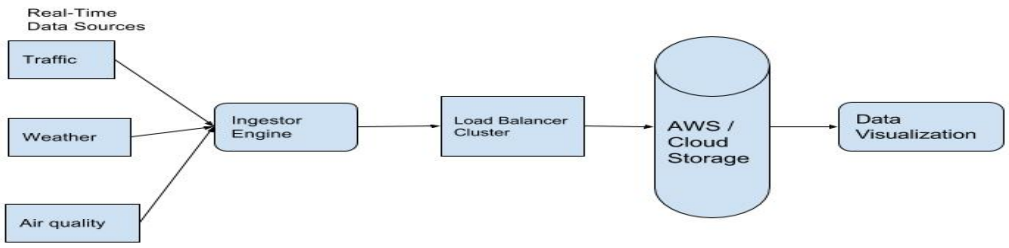

Figure 1: Architecture

The real-time data sources are APIs that that expose publicly available datasets which can be queried through REST calls. Three different sources are used to obtain traffic density/incidents , weather and air quality information in a JSON format.
The Ingestor Engine acts as the 'Publisher' end of the system. The data obtained from the APIs is sent to the Ingestor Engine. The Ingestor Engine is the Producer end of a Kafka queue. Since the rate and size of data burst is unknown, the Ingestor Engine handles it using partitions and replicas. The partitions ensure that the data is distributed evenly across the various nodes for efficiency and scalability.
The Load Balancer cluster acts as the 'Consumer' end of the Ingestor Engine. The Cluster collates the air and weather data into a single JSON document for ease of storage and access. The Load Balancing cluster is the focal point of the project. It uses a hybrid dynamic load balancing technique in order to distribute incoming data to different nodes using a central coordinator (master) node which controls a set of worker nodes. Now the output of the nodes is sent to the AWS cloud using ElasticSearch which hosts all the data persistently. Queries are run in the Data Visualization module which is integrated with AWS/ElasticSearch to display results in a variety of illustrative formats using user queries.

4 Methodology

4.1 Data Extraction

Weather data was extracted using a public API - the open weather map. This API takes as input a city and returns a comprehensive list of weather features for that city. Our method filters it and encapsulates only relevant parameters into a JSON. The parameters include maximum and minimum temperatures , humidity, wind speed and pressure. The Air Quality data is extracted from the National Weather Service. It has only a single parameter of interest - Air Quality Index (AQI). The higher the Air Quality Index, the more polluted the air is for that city.
Traffic data is taken from MapQuest which shows live traffic related data. Traffic data is collected as a set of JSON documents, each of which refers to a traffic incident in a particular city. Each incident is marked with a severity level (0 being least severe and 4 being most severe) along with other parameters such as type of the incident, start time and end$_time$.

4.2 Ingestor Engine

All incoming information is divided into topics. There are three topics - Air, Weather and Traffic. Each topic has a set of partitions for purposes of load balancing. The Air and Weather topics each have 2 partitions and the Traffic topic has 3 partitions on the cluster server. A partitioner class implements the logic to direct incoming JSON to the correct partitions in the kafka topic.

4.3 Load Balancer

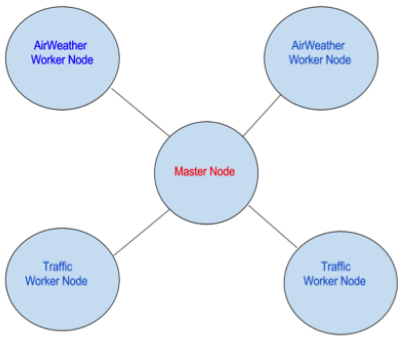

Figure 2: Load Balancing cluster

The above figure depicts our cluster which handles data from the Ingestor. The cluster consists of a master node and several worker nodes. The cluster dynamically load balances itself by adding/deleting worker nodes based on requirement. Since we deal with two different types of messages from kafka, namely trafficJSON and airweatherJSON, there are two types of worker nodes respectively. By separating the worker nodes, we intend to provide a micro service like architecture, in which case, on adding new types of data in the Ingestor engine, the load balancing cluster can easily be scaled up by adding new nodes for the new messages.
The master node periodically queries individual worker nodes for its load rate. The worker node responds by sending its load rate, here load rate implies the number of messages pulled from the Ingestor. The master node maintains a sliding window of size 5 (configurable) for each worker node. It each stores the load rate of the corresponding worker in this window. Every minute the master node queries worker node for load rate in the past minute and adds it to the corresponding window. If the average of all the five load entries i.e if the average load rate in the past five minutes is above a certain threshold, then it means that the worker is

heavily loaded. So master node creates a new worker node for the same topic.
If the average load rate is lesser than a certain minimum threshold, then it means that the worker is underloaded, so master node stops that worker node. We also handle the case, in which, on creating a new worker node, we wait for this new node to actually reduce the load, so this might take some time, hence we do not immediately create another new worker.
The master and worker nodes communicate with each using TCP sockets. On bootup, the worker node, sends its type (traffic or airWeather) to the master. Hence the master node can keep track of each worker node, its load rates and its type. The load rate request and response are also sent through the same TCP socket communication channel.

4.4 Elastic Search and Cloud Storage

All the data from the our cluster has to be stored inorder to run queries on it and visualise the data. We have chosen elasticsearch as our storage end point. The reason for chosing elasticsearch is due to the various analytical queries that it supports which are helpful to visualise the data. We are using the elasticsearch as a service in AWS. The reason for chosing cloud storage over local storage is due to the high volume of data we are dealing with and high availability provided by cloud service. Storing the data on cloud also provides a centralised way of storage so that it can be accessed from anywhere. Also cloud storage helps our need for scalability.

4.5 Visualization

ElasticSearch provides a visualisation platform called Kibana to run analytical queries and visualise the data stored in it. Kibana is an open source plugin for elasticsearch. We have modelled our data as time-series data and Kibana provides different types of visualisation like bar graph, heat map, maps, pie chart etc on such time series data.

5 Scalability and Fault tolerance

Our architecture is designed in a very scalable manner. Since the intended application is visualisation of live streaming data, we can ingest any number of new data into kafka cluster. The load balancing framework is designed in a micro-service architecture. Adding new types of messages to kafka cluster can be scaled up by creating new nodes for the same message and the master node takes it from then on to automatically load balance.

Since the load balancing framework is of master-slave type, if the master node goes down then the cluster is unreliable. Hence, leader election protocol can be implemented to handle this issue.

6 Tools and Environment

1. **Java Maven Framework** Maven allows us to create end-to-end applications using Java by efficiently integrating various libraries and softwares. The weather, air quality and traffic data was obtained using Java's internal HTTP libraries and the data was parsed in JSON using Java's simpleJSON library. The bulk of code is written in Java Programming language.

2. **Kafka** The Ingestor Engine is developed using Kafka. Kafka is a tool used to manage multi-broker message queues in a producer-consumer model. The main reason for using Kafka is its scalability and ease of use. In addition, they have internal support for customised load balancing and load shedding techniques. In order to run Kafka, we need to run the zookeeper and the Kafka server in the host machines.

3. **AWS Elastic Cloud** For implementing a cloud database, we host it on Amazon Web Services(AWS). The advantage of using AWS is that it is well-integrated with ElasticSearch so the ElasticSearch instance can directly be used on the AWS cluster. We are using Elastic Cloud which is a publicly

available tool for managing ElasticSearch queries on AWS cloud. Elastic Cloud is easily configurable and it provides reliable security to cluster data. In addition, Elastic Cloud offers built in integration with Kibana which is the main tool used in visualization of data. The main role of ElasticSearch is to manage data. It also acts as a query engine and provides a framework to query the data.

4. **Kibana** Kibana is a visualization framework that is closely integrated with ElasticSearch. Kibana allows us to query ElasticSearch data into visuals such as pie charts graphs, heat maps etc. One of the main advantages of Kibana is that is seamlessly indexes the entire data which allows it to be queried with minimum delay.

7 Results

Please see Appendix for mapping of city IDs to city names.

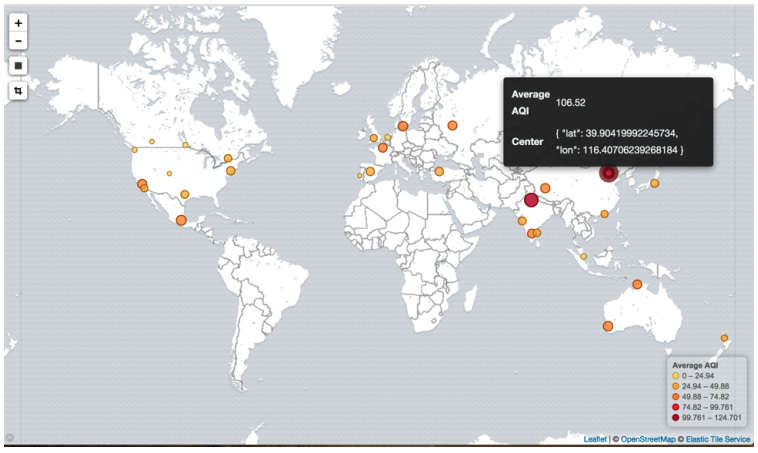

Figure 3: Average Air Quality Index for several cities for the last 7 days

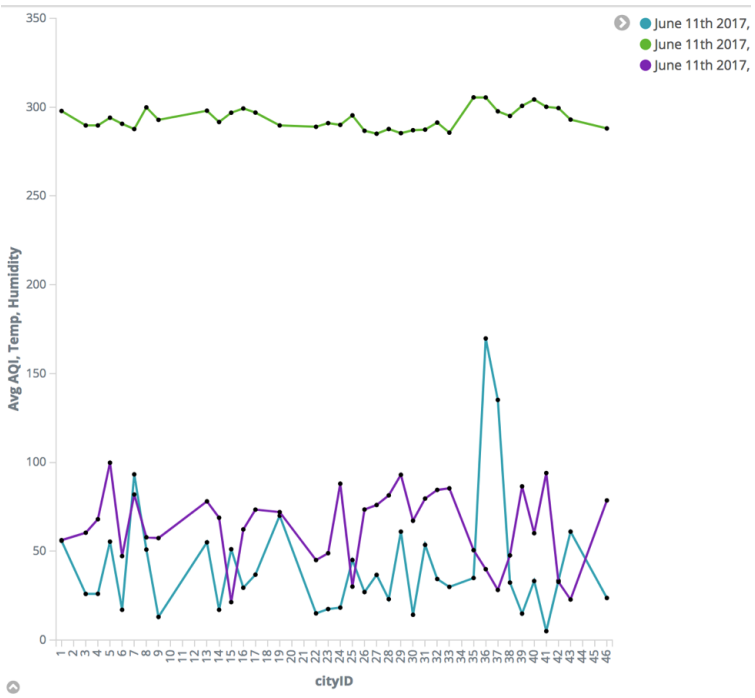

Figure 4: Average AQI, temperature and Humidity for several cities on a single day

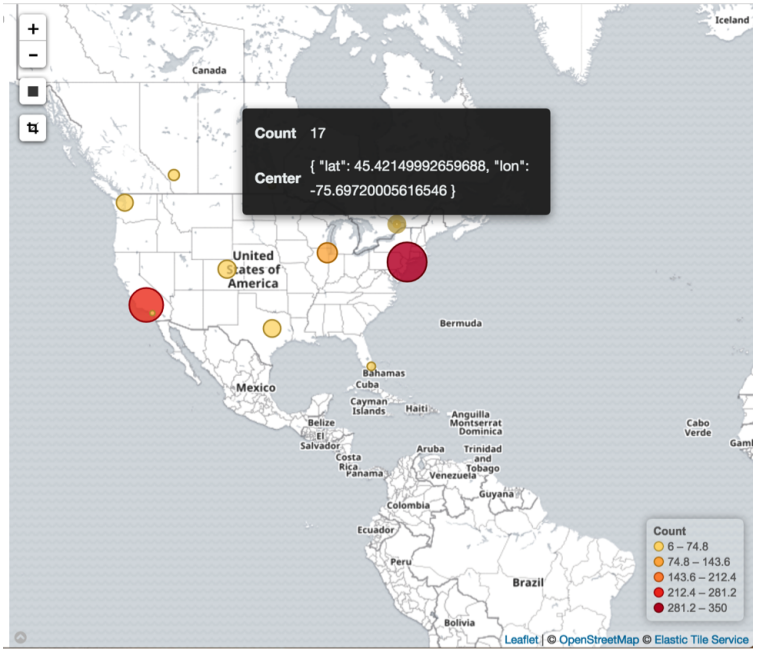

Figure 5: Level of traffic congestion count in various cities in North America

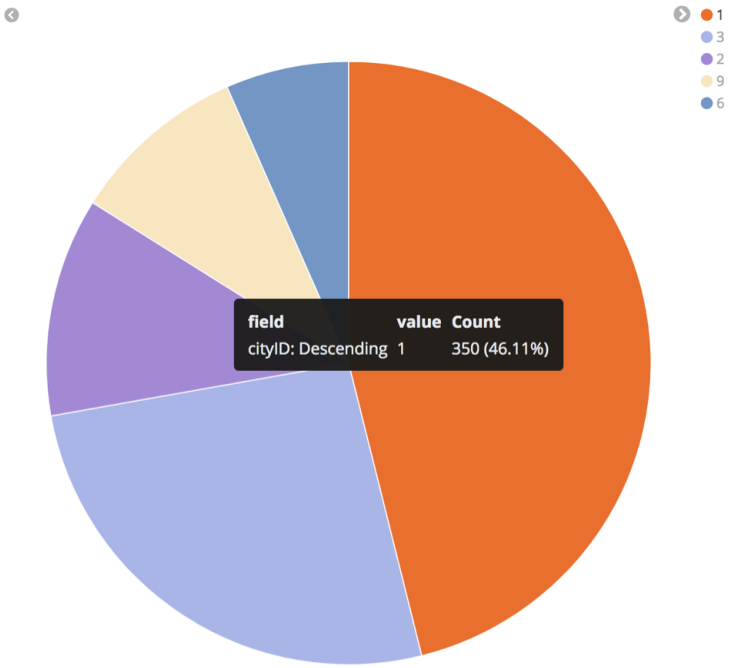

Figure 6: Results for top 5 most polluted cities in the US

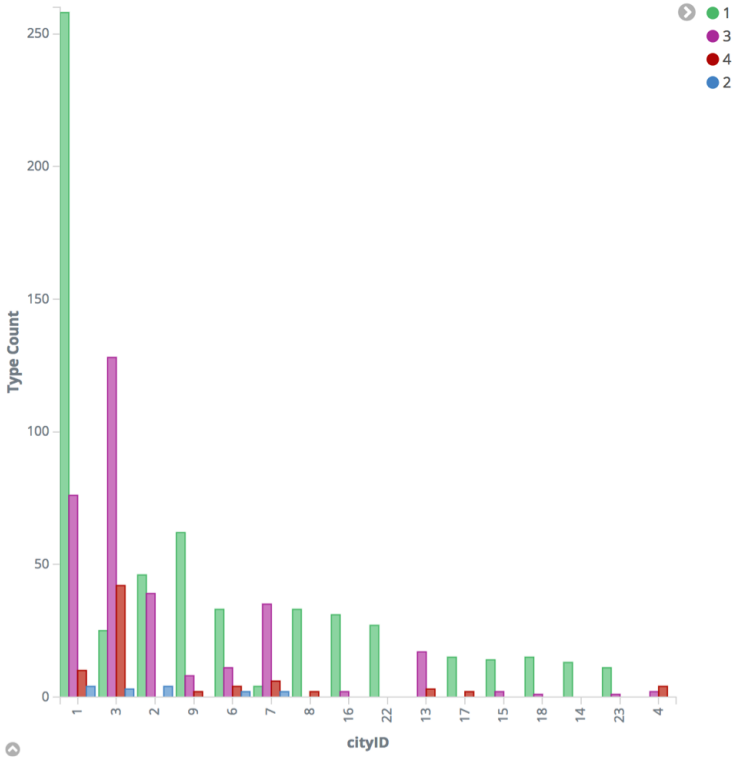

Figure 7: Severity of traffic incidents(0-4) for several cities

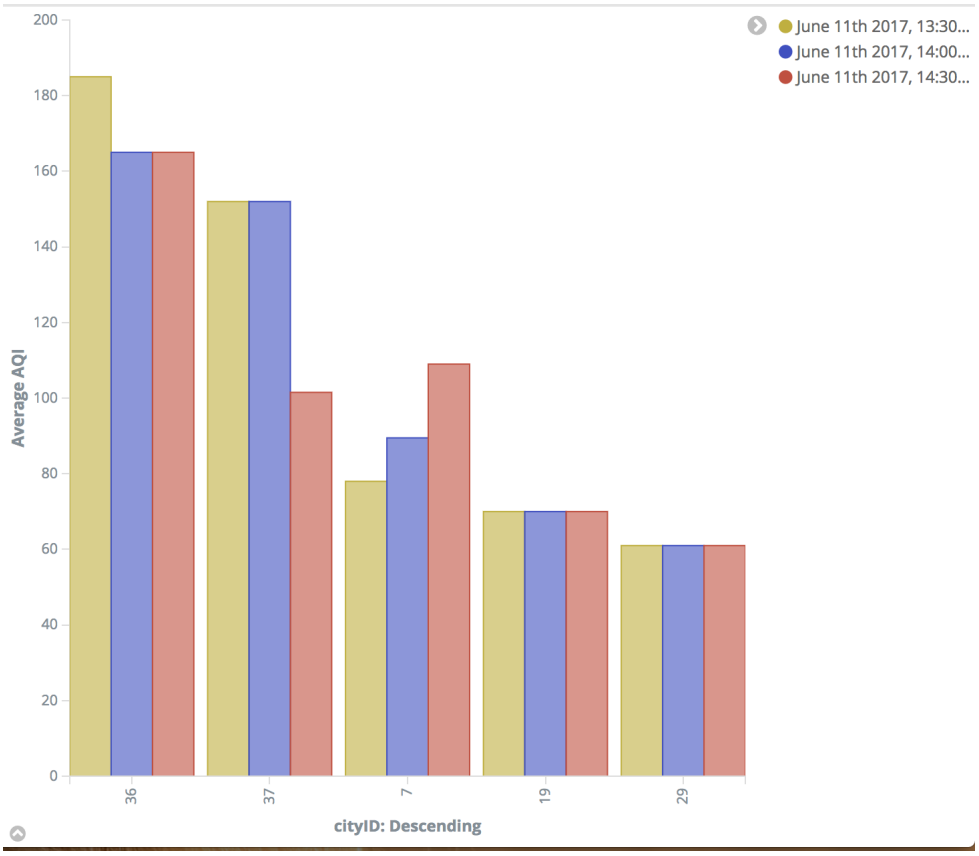

Figure 8: AQI per city basis per hour

8 Future Scope

There are several ways to expand the scope and efficiency of the project.

- Implement a leader election protocol which allows us to make a new master node whenever the current master node fails which would make the system more fault-tolerant.
- Port the entire application to the cloud thus allowing it to be run as a real-time visualization service. This is possible by hosting the entire application on a set of production level servers.
- Connect to a real-time notification system which notifies appropriate agencies if air quality index or traffic incidents exceed a threshold.

9 Conclusion

The application is highly scalable because of the Load Balancing system and ElasticSearch. This allows us to organize large amounts of data and visualize them. Moreover, the scope of data can be vastly extended. We can add more cities and model and visualize complex climate and pollution related data. This project allowed us to learn how to create an end-to-end Middleware system that can handle Big Data. Moreover, this project also allowed us to implement various Middlware and Distributed Systems concepts such as Load Balancing, Message Passing , REST Services etc.

10 References

1. Zhang, Kai, and Stuart Batterman. "Air pollution and health risks due to vehicle traffic." Science of the total Environment 450 (2013): 307-316.

2. Cai, Hao, and Shaodong Xie. 2011. "Traffic-related air pollution modeling during the 2008 Beijing Olympic Games: The effects of an odd-even day traffic restriction scheme." Science of the Total Environment no. 409 (10):1935-1948.

3. Babcock, Brian, Mayur Datar, and Rajeev Motwani. "Load shedding techniques for data stream systems." Proceedings of the 2003 Workshop on Management and Processing of Data Streams. Vol. 577. 2003.

4. Wingerath, Wolfram, et al. "Real-time stream processing for Big Data." it-Information Technology 58.4 (2016): 186-194.

5. Tatbul, Nesime, Uğur Çetintemel, and Stan Zdonik. "Staying fit: Efficient load shedding techniques for distributed stream processing." Proceedings of the 33rd international conference on Very large data bases. VLDB Endowment, 2007.

6. G. Kulkarni, R. Waghmare, R. Palwe, V. Waykule, H. Bankar and K. Koli, "Cloud storage architecture," 2012 7th International Conference on Telecommunication Systems, Services, and Applications (TSSA), Bali, 2012, pp. 76-81

7. Y. Huo, H. Wang, L. Hu and H. Yang, "A Cloud Storage Architecture Model for Data-Intensive Applications," 2011 International Conference on Computer and Management (CAMAN), Wuhan, 2011, pp. 1-4.

8. Vo, Hoang Tam, Chun Chen, and Beng Chin Ooi. "Towards elastic transactional cloud storage with range query support." Proceedings of the VLDB Endowment 3.1-2 (2010): 506-514.

9. Chou, Timothy C. K., and Jacob A. Abraham. "Load balancing in distributed systems." IEEE Transactions on Software Engineering 4 (1982): 401-412.

10. Md. Firoj Ali and Rafiqul Zaman Khan. "The Study On Load Balancing Strategies In Distributed Computing System". International S Journal of Computer Science Engineering Survey (IJCSES) Vol.3, No.2, April 2012

11 Appendix

The Mapping of City IDs to City Names

City Id	City Name	City ID	City Name
1	New York	2	Chicago
3	Los Angeles	4	Irvine
5	Bangalore	6	Seattle
7	San Francisco	8	Austin
9	Denver	10	Paris
11	London	12	Beijing
13	Atlanta	14	Vancouver
15	Las Vegas	16	Dallas
17	Ottawa	18	Miami
19	Mexico City	20	Lima
21	Buenos Aires	22	Calgary
23	Winnipeg	24	Lisbon
25	Madrid	26	Dublin
27	Copenhagen	28	Amsterdam
29	Moscow	30	Istanbul
31	Ankara	32	Dubai
33	Chennai	34	New Delhi
35	Tokyo	36	Hong Kong
37	Singapore	38	Mumbai
39	Darwin	40	Perth
41	Melbourne	42	Auckland
43	Tokyo	44	Hong Kong

Traffic Incident Types

Type	Description
1	Construction
2	Event
3	Congestion/Flow
4	Accident