GridDB Case Study:

DENSO Drive Metrics System

May 8, 2018
Revision 0.10

Introduction

How safe do you drive? It's a relatively hard question to answer since we don't have any measurable and relevant criteria to judge each driver's safety level. If we scored each driver's driving behavior, that would be quite useful for almost all stake holders in this motorized world and would make the world safer place.

A leading supplier of advanced automotive technology, systems and components for major automakers, DENSO International America, invented a technology to quantitatively evaluate a driver's safety level based on machine learning methods. Using this technology, they are building a cloud base solution called the Drive Metrics System (DMS) that allows fleet, rental, insurance organizations, and individual car owners to monitor driving behavior.

As of March 15, 2018, the beta version system of DMS has been completed which is described in detail in this case study.

Challenge

With the DMS software installed in the vehicle, there needed to be a method to stream the driving data, such as velocity, acceleration, location, and current user input to the cloud for real time streaming, archival, and offline analysis. On the cloud side, the receiving data must be written to the database without data loss while concurrently be visualized in a web application in near real-time. The whole system would need to be cost effective during initial field trials and also be able support hundreds of thousands of vehicles storing millions of hours of driving behavior data while maintaining 99.9% uptime.

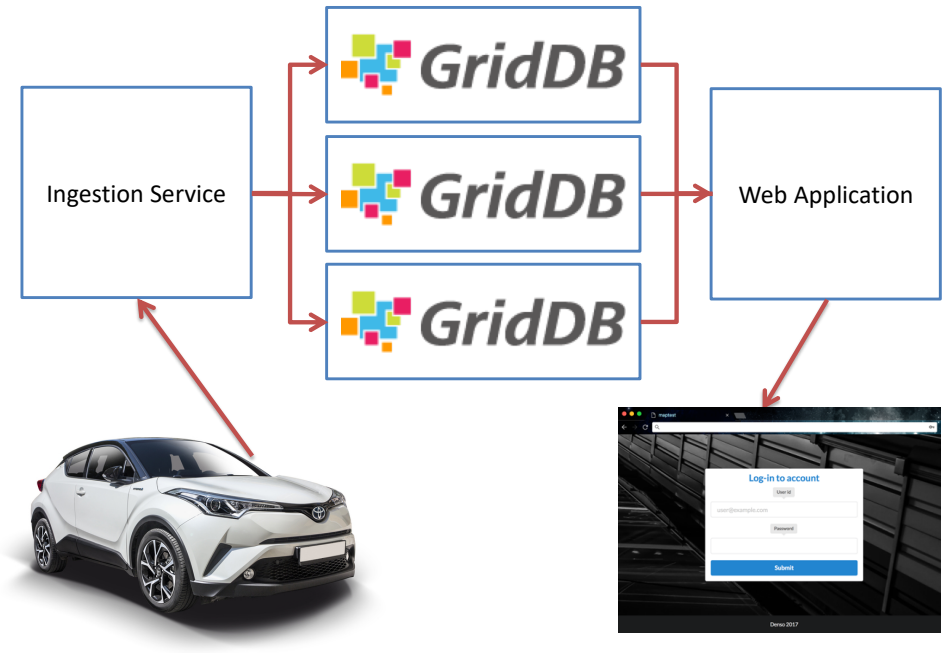

Figure 1. Overview of DMS

Solution

Figure 1 shows the overview of this system. Adding cellular communications allows the vehicle to stream driving behavior data to DENSO's cloud infrastructure where it is processed by an open-source messaging framework before being stored in a 3-node GridDB cluster that could be scaled out as required. The raw input data such as vehicle speed as well as the DMS results can be then visualized in both in near real-time or past trips can be replayed using a web browser.

Results

Figure 2. shows the result of performance evaluation of this system. We measured the average response time between messages from the web server. The web server tries to send a message every 100 msec once a browser accesses to this web server and request to show a trip stored in GridDB. So, in the ideal situation, it should be 100 msec. The left figure shows the average response time depends on the number of browser connections. In this evaluation, the backend server is receiving a data from fixed 1000 vehicles. The right figure shows the average response time depends on the number of vehicle connections. In this evaluation, the frontend server is sending messages to fixed 80 browsers. As you can see, the number of browser connections exceeds 80, the average response time is exponentially increased. It is safe to say that this system can handle around 70 browser accesses at the same time. The vehicle side is more robust. When we connect 1000 vehicles, the delay of each message is still acceptable level. We cannot conduct the larger number of vehicle connection test because of the lack of performance of a workload generator computer. In this evaluation, we used Microsoft Azure instances with following specifications:

- Web Application Server: Standard B1ms (1 vcpu, 2 GB memory)
- Data Ingestion Server: Standard B2s (2 vcpus, 4 GB memory)
- Database Servers: Standard B1ms (1 vcpu, 2 GB memory) x 3

This is a minimum configuration using small instance sizes. Nevertheless, this system is able to maintain high frequency real-time data streams for over 1000 vehicles and 70 browser sessions. GridDB's high performance provided by a hybrid in-memory architecture allowed the ingestion software to support a vehicle for just pennies each while it's Key-Container architecture provided efficient queries making the visualization software easy to develop. The evaluation result also showed that as the number of vehicles generating data increases, the system architecture that will easily scale out without major modification providing while also providing economy of scale that will reduce per-vehicle costs.

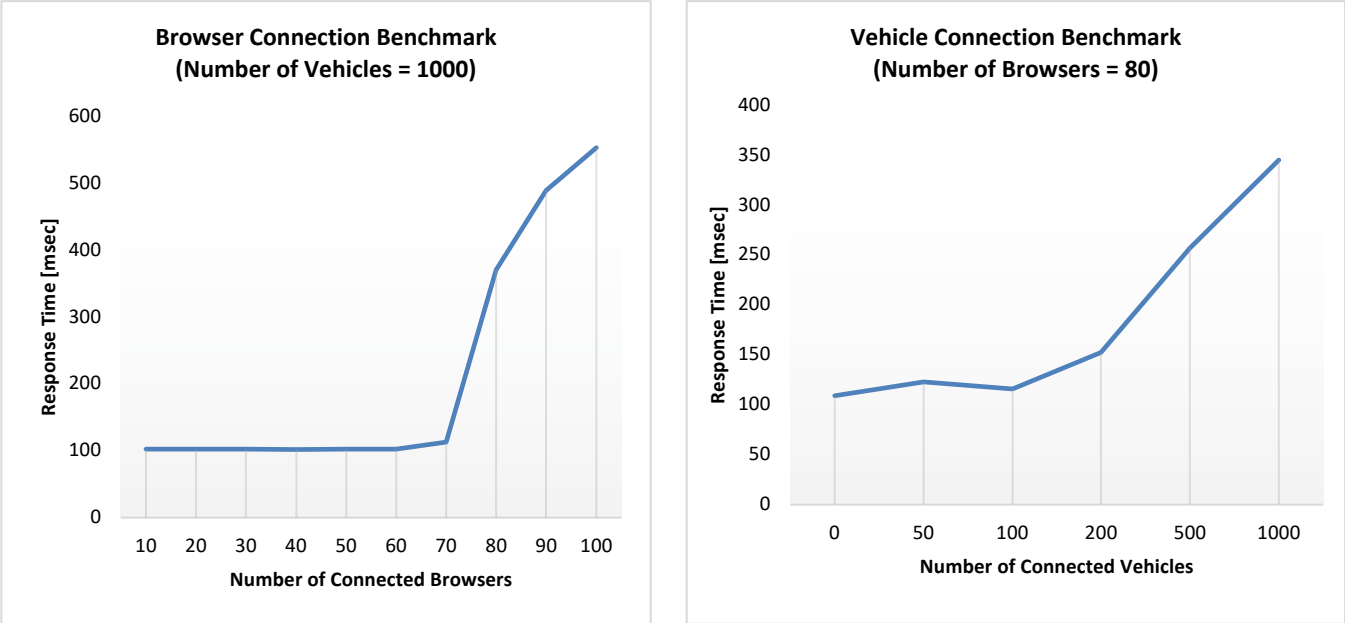

Figure 2. DMS performance result

Future Development

As shown in in Figure 1, the web application server and ingestion server are so-called single points of failure (SPOF), to correct the SPOFs, Azure's load balancer and multiple instances of each application will be deployed. According to Microsoft Azure Service Level Agreement (https://azure.microsoft.com/en-us/support/legal/sla/virtual-machines/v1_0/) , if two same instances are deployed in one "Availability Set", 99.95% uptime for the system is guaranteed.

On the other hand, multiple GridDB database servers have already deployed in one availability group with GridDB's Autonomous Data Distribution Algorithm (ADDA) balancing containers and providing replication allowing the database backend to not only be reliable, but also scalable.

After the completion of performance testing, GridDB was upgraded from Community Edition(CE) to Standard Edition(SE). Table 1 show the differences between CE and SE. From the management point of view, the big difference between CE and SE is Online Backup and Online Expansion. For most databases, adding more nodes can only be done when the database is offline. GridDB Standard Edition offers the ability for users to expand their database online without having to stop operation and halt services. When adding new nodes to a cluster in Community Edition, all the nodes in the cluster must leave the cluster and then be restarted. This can be tedious and costly for a large database with many nodes. In Standard Edition, this step can be skipped altogether.

Another future enhancement is to improve both the accuracy and processing speed of the data analysis. DMS provides several different analysis tools that consist of complicated data search queries and are computationally intensive. As every data point in DMS has both time and geospatial information, GridDB

SE's geospatial optimized queries using POINTS, LINESTRINGs, and POLYGON data types, the development of advanced analysis tools with greater accuracy and lower processing times will be eased.

	CE	SE
SOFTWARE SUPPORT		
Maintenance Releases	n	y
Bug fixes/Patches	n	y
Updates	n	y
BASIC		
Distributed Data Management	y	y
Transaction Management	y	y
DATA TYPE SUPPORT		
Key-value data	y	y
Time-series data	y	y
Geometry data	n	y
QUERY LANGUAGE		
SQL Subset (TQL)	y	y
SCALABILITY		
Offline expansion	y	y
Online expansion	n	y
PERSISTENCY		
In-Memory and Disk	y	y
API		
Java	y	y
C	y	y
ADMINISTRATIVE TOOLS		
Offline backup	n	y
Online backup	n	y
Export/import	n	y
Differential backup function	n	y
Management GUI	n	y
Status Acquisition	n	y

Table 1. Differences between GridDB CE and SE