CarStream： An Industrial System of Big Data Processing for Internet-of-Vehicles

Mingming Zhang, Tianyu Wo, Tao Xie, Xuelian Lin, Yaxiao Liu
zhangmm@Illinois.edu

Outline

- Background of CarStream
  - Business
  - Dataset
  - Challenges
- Solution of CarStream
  - Overview
  - Design and Implement
  - Deploy
- Application Examples
- Experiences

UCAR - Private Car Service

10M users
100+ million orders/year

30,000 cars
50,000 hired drivers

- **Main requirements from the company**
  - ✓ Gather car info in real-time, to track, save gas, etc.
  - ✓ Evaluate/Improve quality of service of each order, e.g. faster pickup
  - ✓ Evaluate/Improve quality of drivers

DataSets

1. Vehicle data: (more than 60 attributes)

Name	Note
Door	Status of four doors and trunk door
Error Code	Error code of engine
Hand-break	Hand break status
Foot-break	Foot break status (%)
Air condition	Air condition control status
Gear position	
Mileage	Total mileage of vehicle
Mileage-accumulate	Mileage accumulated since connector has been connected
Fuel	Remining fuel (L)
Temperature	Engine temperature
Speed	Vehicle speed
RPM	Engine round per minute
Engine oil	Lifetime remained of engine oil
Acc-pedal	Accelerate pedal status (%)
Acc	Acc signal
Light	Light status including full and low headlights, turn lights

From On-Board Diagnostic (OBD)

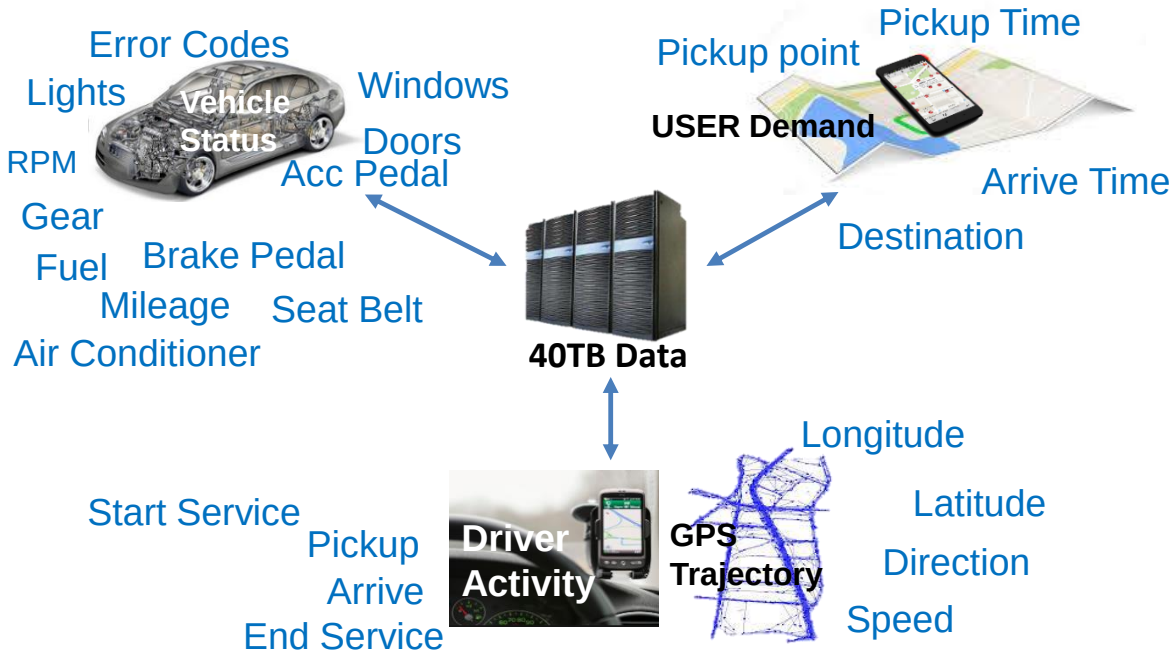
2. GPS data:
latitude, longitude, direction, and speed.

3. Order data:
•<Pickup point, Destination, Start time, End time>
• From user apps.

4. Driver data:
•<start work> <pick up a user> <start service> <stop service> <off duty>.
•From driver app.

Challenges

- Big data: Tens of TB per year. Burst data spikes with traffic peak.
- Scalable: the number of service cars and users are increasing over time.
- Low data quality: Multiple vehicle types, data missing, delay, and disorder widely exists.
- Real time: Quickly extract information from big data with low density of value.
- Reliable.

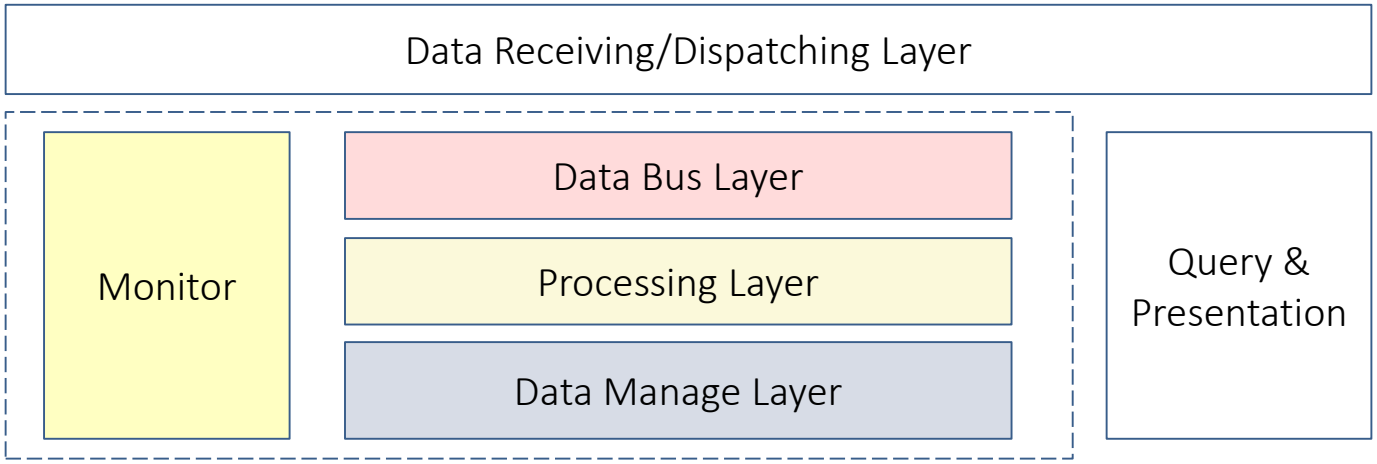


Outline

- Background of CarStream
  - Business
  - Dataset
  - Challenges
- Solution of CarStream
  - Overview
  - Design and Implement
  - Deploy
- Application Examples
- Experiences

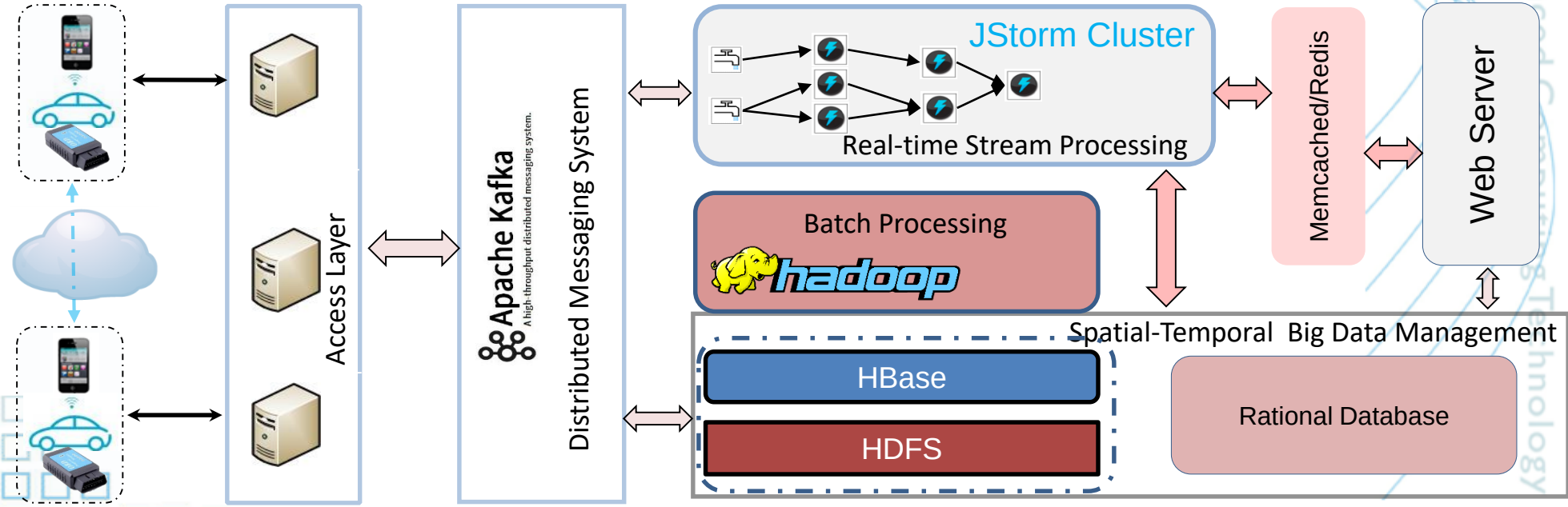
Solution of CarStream

- A multi-layered processing structure:
  - An accessing layer receiving and dispatching data.
  - A data bus layer buffering data and providing data for processing.
  - A processing layer combining stream and batch computing for both online and offline process requirements.
  - A data manage layer for storing and managing data.
- A distributed query & presentation subsystem.
- A monitor subsystem to sense the health status of the system.

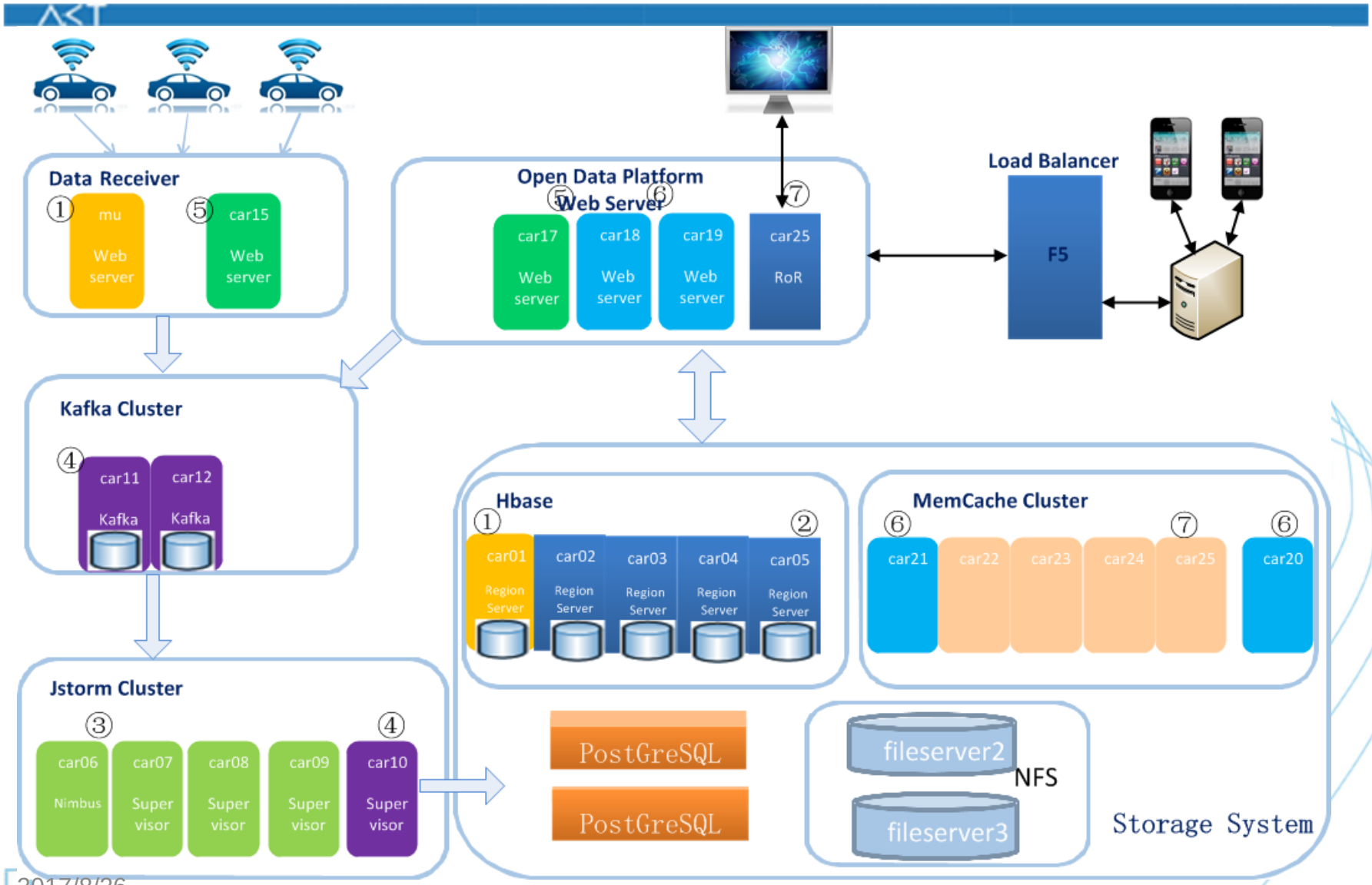


Design & Implementation

- accessing layer: TCP Server
- data bus layer: Apache Kafka as buffering system.
- processing layer
  - JStorm as the stream computation engine. Closely working with Kafka.
  - Hadoop MapReduce as batch processing engine
- data manage layer
  - adopt in-memory caching to accelerate data exchange between webserver and processing engines
  - Archive data into HBase and file system, hot data in-memory and RDBMS.

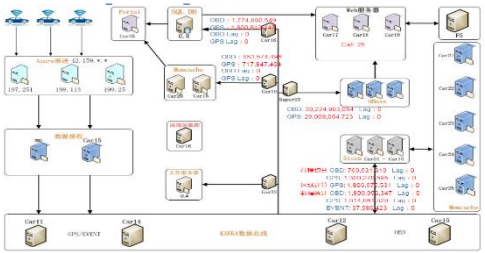


Deployment



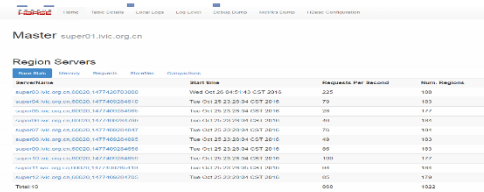
Monitor

Application



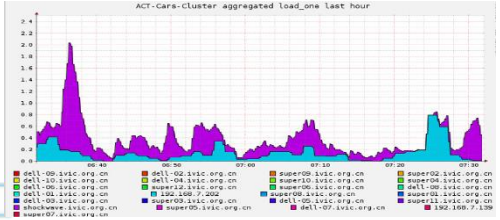
- Consumer Delay
- Topology Throughput Monitoring (Node)
- Data Flow Tracking
- Data Lifecycle Tracking
- Storage Delay

Platform



- TCP Server
- Web Server
- In memory cache
- KAFKA
- JStorm
- Hadoop
- HBase
- Database

Infrastructure Monitoring



- Overall Server Load
- I/O Load
- Memory
- Network Load
- Disk Capacity
- CPU

Outline

- Background of CarStream
  - Business
  - Dataset
  - Challenges
- CarStream Solution
  - Overview
  - Design and Implement
  - Deploy
- Application Examples
- Experiences

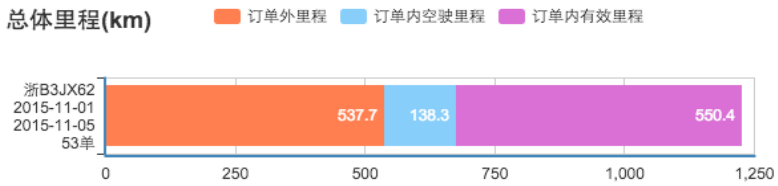
Application Examples - Fleet

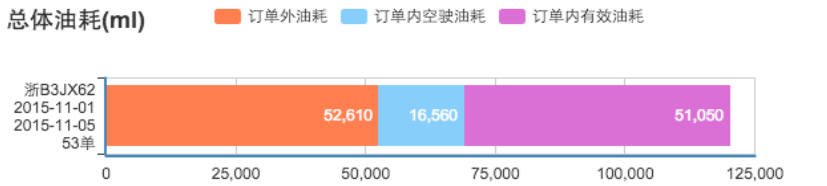
Fleet distribution tracking

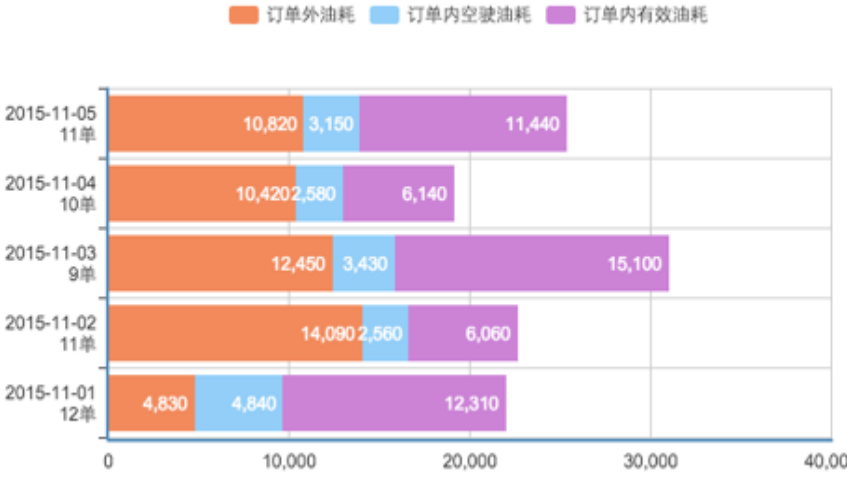
Fleet data statistic

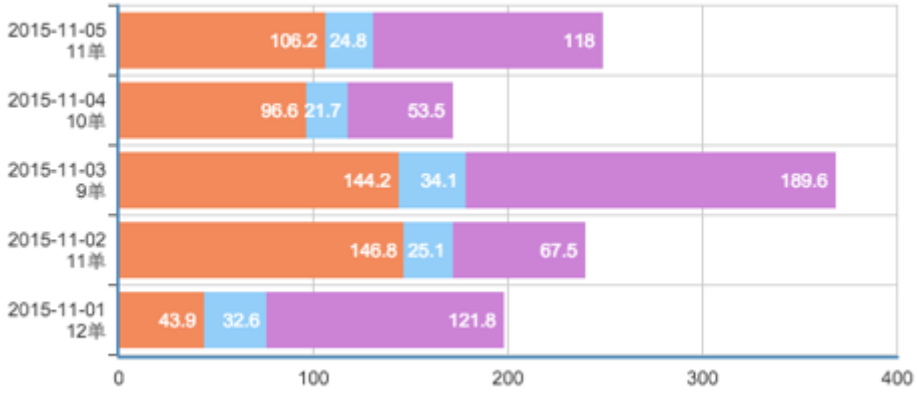
Application Examples - Trip

- Millage classification (with / without passenger)
- Gas stats for individual car/driver/trip





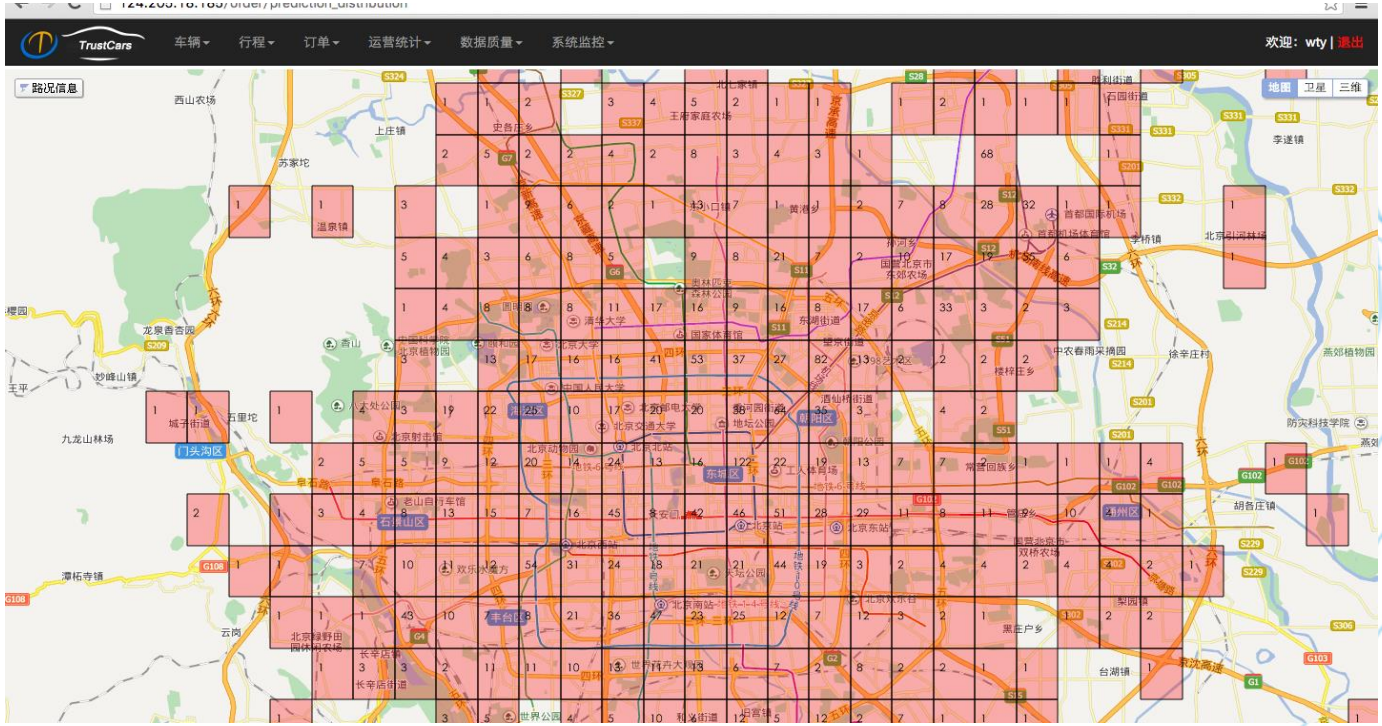




Application Examples - Order prediction

- For driver: show me the potential order in the near 15 mins. Considering other driver's behavior.
- For customer: reduce the waiting time

Order data
＋
GPS data



Outline

- Background of CarStream
  - Business
  - Dataset
  - Challenges
- CarStream Solution
  - Overview
  - Design and Implement
  - Deploy
- Application Examples
- Experiences

A Few Lessons Learned

- Application-level monitoring is necessary for achieving high reliability.
  - We designed a 3-layered architecture to assure the reliability
- A single storage is usually insufficient, and a heterogeneous storage systems is necessary for managing large-scale vehicle data.
  - Separate hot data with archived (cold) data;
  - use in-memory caching as the exchange media for the hot data.
  - Avoid scanning a big dataset by extracting a small dataset using pre-processing.
- Low data quality widely exists in IoV. A fixing model extracted from historical data patterns can be helpful.
  - Based on the data patterns generated from the collected dataset, we design a shared task to improve the data quality.

Data Quality Issues

Problem	Consequence	Causes	Solution
Data Loss	No result/ Inaccurate result	Network Failure	Redundant deployment. Interpolation by data patterns.
		Software fault	
Insufficient Data	Inaccurate result	Physical limitation	Interpolation by data patterns.
Disorder	Wrong result Inaccurate result	Distributed processing	Delay & Wait. Prediction. Order guarantee design in application logic.
		Network delay	
		Store & forward design	
Wrong/Outlier Data	Wrong result	Hardware malfunction	Outlier detection. Data cleaning. Fixing with data patterns.
		Inaccurate sensors	
		Physical problems	

Thanks for listening!