CS428:
Case Study: Software in Big Vehicle Data Processing

Guest Lecturer: Tianyu Wo
Content

- Background
- Requirement analysis
- Architecture design
- Key technologies
- System deployment
- Research topics
- Discussion
Background - Lots of Buzzy Words

- CPS, IoT
- Telematics, Connected Vehicles
- Self-driving, Autonomous car

- Big Data, MapReduce / Spark
- Cloud Computing, Virtualization
Gartner Hype Curves

- Hot (Buzzy) concepts / technologies
Background – Internet of Vehicle

Vehicles

Manufacturer

Communication Service

Customer Service

Maintenance Service

Fleet Operators

Personal

Internet-of-Vehicles (IoV)

Vehicle-to-Vehicle (V-V)
What is IoV

• An integrated ecosystem: cloud as center.

• Key techniques:
  – *Real-time stream computing*
  – *Data integration*
  – *Big data storage and mining*
Chauffeured Car Services

- Chauffeur-Driven Car-on-Demand Service
- New services for commuting, travelling
- Better service / car condition
- Higher cost (or not that high)
- Killing Taxi services?
Chauffeured Car Services

- Sharing Economy - Why does it work?
- The Internet:
  - Matching huge amount of requirement and resource
  - Quick & Low cost

![Uber](uber.png)  ![Lyft](lyft.png)  ![Didi](diidi.png)  ![UCar](ucar.png)
Chauffeured Car Service

- ShenZhou UCAR
  - Uber-like chauffeured car service
  - Used to be a (Hertz-Like) Rental Car Company

- **Asset-Heavy** vs Asset-Light
  - Hiring drivers
  - Owing vehicles
  - More control, less flexibility
Content

● Background
● Requirement analysis
● Architecture design
● Key technologies
● System deployment
● Research topics
● Discussion
Requirement

- U-Car Private Car Services
- How to improve the efficiency/safety of cars
- How to improve the performance of drivers
- Need a strong ICT platform to manage
  - Cars / Drivers / Orders
Requirement and Challenge

• Both customs & service provider:
  – Safe, Steady, Expectable services

• U-CAR like employer
  – Own cars, Hire drivers, Pay gas bills, Dispatch orders

• Can we reduce the running cost while promote our income?
  – Where are my cars? Are they in good condition?
  – Where are my drivers? Are they good?
  – Where are my orders? Can we satisfy our customs?
Requirement and Challenge

• From the company
  – Evaluate/Improve quality of service of each order, e.g. faster pickup,
    (10M users, 110 million orders/yr)
  – Evaluate/Improve quality of driver
    (50000 hired drivers)
  – Gather car info in real-time
    (30000 cars with OBD)
• Challenges (Large Scale, Real-time, Evolving Requirements)
  – Not only a data warehouse

Can you draw a UML Use Case Diagram?
So...

• We need an ICT platform
  – Sensing and collecting data in detail
  – Processing the data in real-time
  – Providing the reliable data product in a flexible way

• Metrics
  – Running cost of the whole fleet
  – Custom waiting time before getting on board
  – More effective in-order mileage, less waste mileage
  – Better gas efficiency
  – Better driver behavior, less accidents
Requirement and Challenge

• Providing safe, steady, expectable services
• Challenges
  – Big fleet, growing fast: 500 -> 25000 in 5 month, 30+cities
  – Continuous service: 7*24
  – Unreliable ICT infrastructure → low data quality
  – Changing business requirement
Requirement Analysis

- What data should be gathered?
  - Vehicle's trajectories, status, orders, ...

- How to gather data from vehicles in real-time?
  - On board sensors + Mobile Internet

- Estimation of data scale?
  - 30000 cars, 0.2HZ (5 seconds interval), 100 byte/msg
  - $0.2 \times 24 \times 60 \times 60 \times 30000 = 518,400,000$ msg / day, 52 GB/day

- How to store and process the data?
  - Fast query, robust data store
  - Transactional SQL DB?
Functions (Planned)

• GPS / OBD data gathering / processing / storage
  – 2 years, normally 15GB/day
• Car position monitor / distribution
  – E-Fence (alert the company when the car is out of certain area)
• Trip QoS evaluation
  – Mileage / gas consumption
• Order prediction
• Data quality stats / system monitor
The Dataset (GPS/OBD/Order)

- Err Codes
- Windows
- Air Conditioner
- RPM
- Gear
- Fuel
- Break Padel
- Milage
- Seat Belt
- Phone GPS
- Start Service
- Pickup
- Driver
- Break
- End

GPS
- Latitude
- Longitude
- Speed
- Direction

OBD
- Pickup point
- Destination

Order
- 31TB Raw Data
The Dataset

- **GPS:** *Position, speed, direction*
- **OBD (On-Board Diagnostic)**
  - *Oil consumption, mileage*
  - *Speed, engine speed, break status*
  - *Door, lights, windows, temperature*
  - *Other engine parameters*
- **Order**
  - *start and end location*
  - *VehicleID, time stamp*

*The Dataset*
Stream Processing
Cloud
Batch Processing
Big Data Management
Fleet Operator
Passenger
OBD
Driver
Content

- Background
- Requirement analysis
- **Architecture design**
- Key technologies
- System deployment
- Research topics
- Discussion
System Architecture - overview

Other user-systems

Access Layer

Data Bus (Buffer)

Stream Computing (Online)  Offline Mining

Storage
System Modules

Collection
- RT Data
- Associated Data
- RT Receive
- Effective Buffering
- Instruction Distribute

Processing
- Stream Process
- Batch Process
- Application Management
- Load Balance
- Resource Allocation

System Monitor

Storage
- Structural Space-Time Data
- Stream Data

Visualization
- GIS Real-time Monitor
- Diagram Display

Open Data Platform
Deployment

• Vehicles & Data
  – 25000 vehicles online and still increasing (60000 vehicles expected)
  – 120 million data packages received per day

• Hardware Resource
  – Microsoft Azure 20 VMs
  – Beihang University
    – Computing cluster, 25 servers, 4 cores and 16GB Memory, Bigdata cluster, 64 servers, 400TB
Deployment

Data Receiver

- meu Web server
- car15 Web server

Kafka Cluster

- car11 Kafka
- car12 Kafka

Jstorm Cluster

- car06 Nimbus
- car07 Supervisor
- car08 Supervisor
- car09 Supervisor
- car10 Supervisor

Open Data Platform Web Server

- car17 Web server
- car18 Web server
- car19 Web server
- car25 ROR

Load Balancer F5

Hbase

- car01 Region Server
- car02 Region Server
- car03 Region Server
- car04 Region Server
- car05 Region Server

MemCache Cluster

- car21
- car22
- car23
- car24
- car25
- car20

PostgreSQL

- fileserver2

PostgreSQL

- fileserver3

Storage System

NFS
Data Processing Functions

• Real Time Stream Processing System
  – Data filtering & integration
  – Data quality analysis
  – Anomaly detection
  – Route segmentation
  – Statistics for billing
  – Driving event analyze
  – Driving ability ranking
  – Electronic fence

• Offline Mining
  – Order forecasting
Portal

- Fleet Distribution
  - running track
  - driving event alarm
- Routes Management
- System Monitor
  - application layer
  - system layer
- Statistics
Vehicles Distribution
Route Segmentation & Monitor
Cluster Monitor
Dataflow Monitor
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
Trip QoS evaluation

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

### 总体里程(km)

<table>
<thead>
<tr>
<th>日期</th>
<th>订单外里程</th>
<th>订单内空驶里程</th>
<th>订单内有效里程</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-11-03</td>
<td>105.2</td>
<td>24.8</td>
<td>11.9</td>
</tr>
<tr>
<td>2015-11-04</td>
<td>96.6</td>
<td>21.7</td>
<td>53.5</td>
</tr>
<tr>
<td>2015-11-05</td>
<td>144.2</td>
<td>34.1</td>
<td>189.6</td>
</tr>
</tbody>
</table>

### 总体油耗(ml)

<table>
<thead>
<tr>
<th>日期</th>
<th>订单外油耗</th>
<th>订单内空驶油耗</th>
<th>订单内有效油耗</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-11-03</td>
<td>10,820</td>
<td>3,150</td>
<td>11,440</td>
</tr>
<tr>
<td>2015-11-04</td>
<td>10,420</td>
<td>2,580</td>
<td>6,140</td>
</tr>
<tr>
<td>2015-11-05</td>
<td>12,450</td>
<td>3,430</td>
<td>15,100</td>
</tr>
</tbody>
</table>

### 详细信息 (浙B3JX62)
Content

- Background
- Requirement analysis
- Architecture design
- Key technologies
- System deployment
- Research topics
- Discussion
Key Technologies

- (Onboard) Data Collection
- Data Bus / Data Storage
- Online Processing vs Offline Processing
- Data Visualization
Onboard Data Collection

- Data Preprocessing
- Transmit on demand
- On-board Diagnosed

Caution!
Data Bus

- Cache received data (several days)
- Multiplex (Feed to multiple downstream processing units)
- Efficient (High throughput)
- Robust

Apache Kafka

A high-throughput distributed messaging system.
Apache Kafka

- **High Throughput Message Publish-Consume system**
  - Developed by LinkedIn, Apache Open Source Project (2011)
  - Programmed in Scala, Running on JVM, Provide Java API
  - Usage: Data buffer
    - Different speed: data producer vs consumer
    - Different sources
    - Different business
  - Data persistence: Queue rotation
Apache Kafka

● High Performance
  ○ Data Persistence complexity: $O(1)$
  ○ Thousands of client per node
  ○ $> 100$ MB/s per single node

● Robust
  ○ Distributed Replication

● Scalability
  ○ Auto scaling (Adding new machine without shutdown)
  ○ Partition based scaling
Kafka Architecture

- Kafka Cluster: group of Broker
- Managed by zookeeper
Kafka Concepts

• Broker
  • Kafka Server (cluster node), storing data, identical role in cluster

• Topic
  • Message categories, e.g. GPS, OBD
  • Different topic stored separately, messages of a topic stored on one or more broker(s)

• Partition
  • Partition of topic (physically divided), assigned when creating topic
  • Sequential Append only, ID: unique in one partition

• Replication
  • Replica of Partition, assigned when creating topic
  • leader-follower
Kafka in UCAR system

- Caching data
- Pump data into different storage
Data Storage - Design Consideration

● Package storm
  ● 120 million packages per day, unbalanced
● System reliability, Network reliability
● Data value: Index? Storage? Query?
  ● Over 99% queries focused on the latest data.
  ● But we still need historical data
● Data flow redundancy and system flexibility
● Data visualization
Data Storage - Hierarchical Design

- Multi-layer Storage

Current status

One Month

History Data
Hadoop History

• Google GFS/MapReduce @2003/2004
  – The Google File System
  – MapReduce: Simplified Data Processing on Large Clusters
  – Bigtable: A Distributed Storage System for Structured Data

• MRv1 @2006
  – Doug Cutting

• MRv2/YARN @2013
HDFS - A distributed file system

- **NameNode**: HDFS name system, file-chunk mapping
- **Secondary NameNode**: High reliability
- **DataNode**: data storage
Hadoop MRv2 (YARN)

- YARN Changes
  - JobTracker become ResourceManager (RM) and ApplicationMaster (AM)
YARN components

• ResourceManager(RM)
  – Monitoring ApplicationMaster
  – Resource monitor and allocation

• ApplicationMaster(AM)
  – Partitioning data
  – Applying resource
  – Task monitor

• NodeManager
  – Manage Resource on single (physical) node
  – Execute command from RM
  – Execute command from AM
YARN Container

- Resource Abstraction (CPU/Memory)
- AM apply containers from RM
- Allocated by Scheduler in RM
- Container created by AM, managed by NM
- AM is also running in a container
YARN workflow
HBASE

• Index system of Key-Values, structural data
• Scan table

• GPS / OBD / Events / Orders
Hadoop scenario

- Huge file, huge amount of files
- Offline batch processing
- No good for iteration

Use cases
- Log analysis
- Data mining
Hadoop in UCAR system

- Dump history data (in HBASE)
- Offline statistic jobs
  - Order prediction model training
  - Trip QoS stat
JStorm - online data processing

• Alibaba JStorm is an enterprise fast and stable streaming process engine. (Apache Storm - like)

• JStorm: real-time computing
  • Like Hadoop MR: Users write data processing unit and deploy it
  • Run for ever
Online vs offline

• Message driven computing
• How to model your computing task? Think in data stream.

• JStorm contain the abstraction of stream, which is an ongoing continuous unbounded tuple sequence.
Topology

• Topology is the highest abstraction in JStorm, it can be submitted to JStorm cluster to be run there.
  
  • A topology is a data stream conversion chart, each node in graph above represents a spout or bolt, and each edge in the graph represents one data stream. After spout or bolt sends tuple to a stream, it will send tuple to each bolt subscribed from this stream.

  • For real-time calculation, we need to design a topology diagram and implement the handle detail of bolt
Spout/Bolt

- “Spout” and “Bolt” are two basic primitives Storm provides for stream transformations. They have interfaces to be implemented before you can run your application-specific logic.
Tuple

• The data in stream is abstracted as tuple in JStorm. A tuple is a list of values, each value in the list has a name, and the value can be a basic type, character type, byte array, of course, also can be any other serializable type. Each node in the topology of the field in which it must explain emitted tuple name, other nodes only need to subscribe to the name.
Worker/Task

- Worker and Task are the execution units in JStorm, a worker represents a process, a task represents a thread, and tasks run in worker, one worker can run more than one task.
Resource Slot

- In JStorm, the resource types are divided into three dimensions, CPU, Memory, and Port
  - A worker consumes a Port Slot.
  - Topology can set how many CPU slot will one worker cost. If there are some tasks cost too much cpu, please allocate more CPU slot to them.
  - Topology can set how many Memory will be assigned to one worker task. The default size is 2GB. Please enlarge it when not enough.
JStorm Architecture

- Similar to Hadoop, Master / Slave architecture (Nimbus/Supervisor)
- Fault tolerance
Scenario of JStorm

- Highly parallel real-time tasks
- Stateless (Why?)
  - Processing unit depends only on input Tuples
  - No dependency between different data streams
- Use cases:
  - Log Analysis
  - Pipeline processing
  - Data Statistic
JStorm in UCAR

- Distribution of vehicles / Space Aggregation of vehicles
- E-fence
- Trip recognize
- Data quality stat
- ……

Out-of-bound state storage

- Data distribution update
Visualization

- Web Portal
- Evolving requirements
- Fast develop and deploy

- Solution: Ruby on Rails
Why Ruby on Rails?

• Agile Web Dev
  – Philosophy: less code, DRY = Don’t Repeat Yourself
  – Functional programming
  – O-R Mapping (that’s why SQL DB is used)

E.g. one-line histogram stat

```ruby
a=[1,2,3,4,5,1,2,3,7,3]
a.inject(Hash.new(0)) { |hash, x| hash[x] += 1; hash}
```

Output: `{1=>2, 2=>2, 3=>3, 4=>1, 5=>1, 7=>1}`
Content

- Background
- Requirement analysis
- Architecture design
- Key technologies
- System deployment
- Research topics
- Discussion
System deployment

• Combine different system components together
  – VM based deployment (Why?)
    • Improve resource utilization
    • Auto-scale (copy and go)

• Monitor and failure recovery
  – Hardware / Middleware / Application
  – Auto start when booting
  – Eliminate single point of failure
Nimbus

Tomcat
Zookeeper
TCP_Server
JStorm_supervisor
Tomcat JStorm WebUI 8080
OffsetGetter Web Kafka Web监控 8086
Kafka
Mem cache (latest)
monior
Web Service
dataporter_pg GPS
Event From KFK to HBASF
Dataporter mc
Dataporter pg OBD
Event Dataporter 2.0

super03
192.168.0.8
192.168.0.4

Portal 80
CROPN_TAB rrd update (ruby)

DataReceive r2.0.
RouteForward d_2.3
OffsetGetter Web * 2
Monitor_Kafka_1020
OrderConsumer2.0

OrderConsumer

routeComsumer

tomcat

RouteMatcher 4.3
EventGP SMMathc
Content

- Background
- Requirement analysis
- Architecture design
- Key technologies
- System deployment
- Research topics
- Discussion
Research Topics

• Eco-Driving, sustainable transportation
• Spatio-temporal data management
• Orders forecast & Fleet Scheduling Algorithm
• Large-scale moving objects management
• Driving Behavior Analysis
Passenger Demand (Order) Prediction
Devide history order data from both spacial and temporal dimensional. Use this discreted data to train a neural network to predict order number of a certain area (grid).

Online Anomaly Detection for Driving Behaviors
Extract a states graph from huge data set to depict the patterns of how data changes while driving, and use this graph as a metrics to detect anomaly from online driving data stream.

Trajectory Compression
Based on road network and frequent trajectory pattern mining. We first represent location in a <roadid, offset> tuple. Then we use frequent patterns(id) to compress trajectories.
Problems 1: Data Quality

• Missing value filling for streaming data in IoV/IoT scenarios.
  – Motivation: due to many reasons like the network instability or system failure, the uploading stream data might existing missing fragment. Thus requires to predict and fill those missing parts. Based on the historical data, we can achieve this according to similar data evaluation (e.g., similar vehicle status or special-temporal condition)

• Key research point:
  – A status-similarity based missing value/segment predict model
  – Comprehensive effectiveness evaluation of the model (for a certain level of data missing, to what extent this damage can be fixed)
Problems 2: Data Mining

• Carry-passenger predict based on driving behavior analysis.
  – Motivation: Some drivers may use the cars, which belong to the company, to personal use. For example, a driver of UCAR may secretly service passenger from other application platforms other than UCAR, so that they can earn all the service fee. We want to identify this kind of behavior by identify the difference between driving alone and with passengers. We assume that when driving with passengers, the driver would perform better (to be verified). We can use the order data of UCAR as ground truth to classify the driving data (OBD, GPS) into two categories: driving alone or not.

• Key research point:
  – Investigate some data, verify whether there is a difference between the driving behaviors of driving alone and carrying passengers.
  – If the assumption, we need to provide a classification model to isolate the driving period with passengers.
  – By fusing with order data, we can further isolate the abnormal driving period which the driver is probably doing private service.
  – Comprehensive evaluation of the model.
Problems 3: Safe driving

- Spatial anomaly detection with big driving data.
  - Motivation: Transportation anomaly, such as congestion, accidents and road construction, would bring inconvenient for drivers and other participants. It is significant to detect those anomaly so that we can alert drivers or road maintenance department. A detecting solution is by analyze driving behaviors, when anomaly occurs, drivers’ driving style changes. By merging multi drivers’ driving information in real time, we might be able to detect the spatial temporal anomaly in real time.

- Key research point:
  - Anomaly detection oriented driving behavior analysis model.
  - Spatial anomaly detection model
  - Solve the data sparsity problem. We don’t have enough vehicles covering every road segment.
Future work: Open Data Initiatives

• How to provide the data collected/processed to customs / 3rd parties (Government)?
  – Traffic aware navigation, Autopilot
  – Application integration (Service recommendation, Mashup)

• Data Privacy
  – Hiding custom trajectories

• BigData format / interface standards
  – Large original data (large in quantity)
  – Business related derivative data (large in business types)
  – Efficiency in data exchange (compression?)
Content

- Background
- Requirement analysis
- Architecture design
- Key technologies
- System deployment
- Research topics
- Discussion
Discussions

● Robustness
  ○ Redundant Design
  ○ System Monitoring

● Performance
  ○ Throughput
  ○ Delay

● Scalability
  ○ How about 10 million connected vehicles?

● Data integration
Discussions

- System Development and Deployment
  - VM based approach
  - Cloud based approach
  - Container based approach
- Version Control
  - Source code
  - Production deployment
  - Rollback
- DevOps: Development and Operations

[Diagram showing the intersection of Development, QA (Quality Assurance), and Operations, with a DevOps overlap.]

https://zh.wikipedia.org/wiki/DevOps
Q&A

Tianyu Wo
wotianyu@illinois.edu
Beihang University