DATA MODELING FOR IOT

APACHECON IOT NORTH AMERICA 2017

PRESENTED BY:
JAYESH THAKRAR
SENIOR SOFTWARE ENGINEER
WHY DATA MODELING FOR IOT?

1. IoT is the next big wave after social media (e.g. connected cars, smart homes & appliances)

2. Interesting challenges of volume, velocity and variety

3. Can be applied to other big data problems
DATA MODELING FOR IOT

1. Discuss sample IoT application

2. Discuss data model

3. Discuss application architecture
Sample Application
INTELLIGENT VEHICLES

Cloud (Internet)

Road-side infrastructure

Communication Endpoints:

- V2V: Vehicle to Vehicle
- V2C: Vehicle to Cloud
- V2I: Vehicle to Infrastructure
- Event = single, discrete communication message exchanged between a vehicle and infrastructure
V2I: DATA & APPLICATION ASSUMPTIONS

• 1+ billion vehicles
• 500+ events per vehicle/day, based on
  avg. time on road = 3 hours = 180 min
  1 event per 10-30 seconds (avg = 3 per min) = 180*3 = 540 events/vehicle
• Avg. event size = 250-500+ bytes
  Total raw data size = 150-300 TB / day
• Cassandra datastore
  can be applied to HBase or other similarly scalable datastore with appropriate testing
• Streaming for ingestion/processing/ETL
• Adhoc and batched analytics, extraction, etc
• Avoid schema-level indexes
  for maintainability, efficiency, size, storage, etc.
SAMPLE APPLICATION ARCHITECTURE

Ingestion pipeline

- Rcvr
- Kafka
- Processor
- Kafka

Data storage

- Writer
- Cassandra

Stream processing and analytics

Downstream processing

Infrastructure Network
DATA MODEL CONSTRAINTS / REQUIREMENTS

• Efficient, low-latency writes and reads

• Sample queries:
  - Events for a vehicle between two dates (or timestamps)
  - Events for an infrastructure between two dates (or timestamps)
  - Events by all infrastructure on a specific road-segment in a region

• Short, adhoc query characteristics/needs (guesstimate)
  - volume = 100 – 100,000 rows
  - response time = 100 ms – 100 seconds (proportional to result size)
SCHEMA VISUALIZATION: STAR SCHEMA

- Vehicle
- Region
- Event
- Road Segment
- Time / Calendar
- Infrastructure
CAN ALSO BE APPLIED TO: ADVERTISING/SEARCH
CN ALSO BE APPLIED TO: SOCIAL NETWORKS
IoT Data Model
INSPIRATION: UNIX FILESYSTEM INODE

Inode:
- Direct block pointer 1
- Direct block pointer 2
- ....
- Direct block pointer 9
- Direct block pointer 10
- Indirect block pointer 1
- Indirect block pointer 2
- Indirect block pointer 3

Data Blocks:
- Data Block
- Pointers
- Data Block
- Pointers
- Data Block
- Pointers
- Data Block
- Pointers
- Data Block
CASSANDRA: TABLE BASICS

• Data stored in tables with pre-defined schema

• Data types: primitives, collections, user-defined type
  – Collections = sets, maps, lists
  – Map keys and set and list values sorted

• Every table has primary key (PK)
  – PK = single column or multi-column (composite)
  – Data distributed on cluster nodes based on hash of first part of PK

• Keyspace = collection of (related) tables

• PK based queries = very fast
  because of bloom filter, key cache, sstable indexes
# DATA ASSUMPTIONS (SIMPLISTIC MODEL)

## Event Data

| Id | Data = timestamp, infrastructure-id, vehicle-id, speed, direction, location, ..... |

## Vehicle Data

| Id | Data = make, model, vin (hashed?), .... |

## Infrastructure Data

| Id | Data = Infrastructure type, location, road-segment, .... |
TABLE SCHEMA OPTIONS

Traditional table structure - column for each field

\[
\text{INSERT INTO event(id, timestamp, vehicle\_id, infra\_id,...)}
\]

\[
\text{INSERT INTO event JSON '{"id" : 1234, "timestamp" : "...", ....) }
\]

All data fields serialized into a single column

\[
\text{INSERT INTO event(id, data)}
\]
\[
\text{VALUES (1234, "JSON/blob/serialized avro/etc") // data = blob or text}
\]

All data field stored into a collection field (e.g. map and/or set)

\[
\text{INSERT INTO event(id, data)}
\]
\[
\text{VALUES (1234, \{"timestamp": ...\}) // data = map<text, text>}
\]
STAR SCHEMA: DIMENSION TABLES
**veh_event**
Table in-lieu of index on event to access all events for a vehicle in a chronological order

**infra_event**
Table in-lieu of index on event to access all events for an infrastructure in a chronological order

**loc_infra_event**
Table in-lieu of index on event to access all events for a region/location by infrastructure and time
VEHICLE -> EVENTS : VEH_EVENT

CREATE TABLE veh_event(id TEXT PRIMARY KEY, map_data MAP <TEXT, TEXT>, set_data SET <TEXT>, ...)

vehicle_id = eb5071d8-0e35-4a82-ad37-543d3da66de7
event_id = 25b6a3f4-5eec-4b04-954e-6d6bf85c4776

Level 0: Map of pointers to hourly data for each vehicle

| eb5071d8-0e35-4a82-ad37-543d3da66de7 | set_data: (2017062408, 2017062409, ...)
|

Level 1: Map of pointers to actual event data for a vehicle for a given hour interval

| eb5071d8-0e35-4a82-ad37-543d3da66de7, 2017062408 | map_data: (08:23:16.732 -> 25b6a3f4-5eec-4b04-954e-6d6bf85c4776, ...)
|

Actual event data

| 25b6a3f4-5eec-4b04-954e-6d6bf85c4776 | data : ...... |
INFRASTRUCTURE -> EVENTS: INFRA_EVENT

CREATE TABLE infra_event(id text PRIMARY KEY, map_data MAP <TEXT, TEXT>, set_data SET <TEXT>, ...)

infra_id = ffe0bdbb-3b89-4337-a477-4a17f719b559
vehicle_id = eb5071d8-0e35-4a82-ad37-543d3da66de7
event_id = 25b6a3f4-5eec-4b04-954e-6d6bf85c4776

Level 0: Map of pointers to hourly data for each infrastructure

L0, ffe0bdbb-3b89-4337-a477-4a17f719b559 | set_data: (2017062408, 2017062409, ...)

Level 1: Map of pointers to actual event data by vehicle for an infrastructure for a given hour interval

L1, ffe0bdbb-3b89-4337-a477-4a17f719b559, 2017062408 | map_data: (23:16.732, eb5071d8-0e35-4a82-ad37-543d3da66de7 -> 25b6a3f4-5eec-4b04-954e-6d6bf85c4776, ...)

Actual event data

25b6a3f4-5eec-4b04-954e-6d6bf85c4776 | data : ......
LOCATION -> EVENTS: LOC_INFRA_EVENT

CREATE TABLE loc_infra_event(id text PRIMARY KEY, map_data MAP <TEXT, TEXT>, set_data SET <TEXT>, ...)

region_id = 3aa40699-357e-48db-888b-af2ff7856949
road_seg_id = 60b57655-0670-4969-9eec-99bcf8c8a034
infra_id = ffe0bdbb-3b89-4337-a477-4a17f719b559

Level 0: Map of pointers to road-segments by region

| 3aa40699-357e-48db-888b-af2ff7856949 | set_data: (60b57655-0670-4969-9eec-99bcf8c8a034, ...) |

Level 1: Map of pointers to infrastructure by road-segment

| 60b57655-0670-4969-9eec-99bcf8c8a034 | set_data: (ffe0bdbb-3b89-4337-a477-4a17f719b559, ...) |

map_data can be used above if there is a need to store any data (e.g. timestamp) along with road-segment or infra id.
LOGICAL & PHYSICAL DESIGN CONSIDERATIONS

- Split each "level" of (logical) event navigation table into physical tables
  - E.g. vehicle_event into vehicle_event_lo, vehicle_event_l1
    Allows tuning parameters like cache, partition size, bloom filter as well as maintenance, etc.

- Primary keys for tables – combine process-level UUID + counter E.g.
  - <uuid>-<NNNN> (reduces number of UUID generation calls)
  - Further compact primary key by using binary encoding instead of string
    (e.g. 16 bytes for UUID + 8 bytes for counter)

- Short column names and appropriate data formats
  - CREATE TABLE vehicle_event(id BLOB PRIMARY KEY, m MAP <TEXT, TEXT>, s SET <TEXT>, ...)
  - Compact data e.g. time-of-day timestamps as integer i.e. ms of the day)

- Data immutability (helps reduce Cassandra entropy & ghost data concerns)
  - Immutable event level data (insert-only into event and navigation tables)
  - TTL to "age-out/purge" old data

- Keyspace sharding by time period and Cassandra compaction strategy
  - Keyspace by day/hour Compaction strategies - LCS, STCS and DTCS/TWCS
KEY TAKEAWAYS OF DATA MODEL

- Single column primary keys
- Short primary key and column names
- All access (single row or range scan) via primary keys only
- Range scan (when necessary) appropriately paginated
- Immutable data (no updates/deletes) and idempotent inserts
- Data purge (TTL v/s keyspace by time period)
The Big Picture
Data Architecture + App Architecture
SINGLE CLUSTER, CENTRALIZED INGESTION & PROCESSING

Single, centralized Cassandra cluster with data-pipeline from different locations
MULTI-DATACENTER CLUSTER, INGESTION & PROCESSING

Dist. Cluster across multiple datacenters
MULTIPLE INDEPENDENT, MODULAR SYSTEMS

Multiple, independent Cassandra clusters at different datacenters along with an optional central cluster containing select and/or aggregated data.
Reference & Misc
SAMPLE OF V2I REFERENCE INFORMATION

- https://www.its.dot.gov/index.htm
- https://www.its.dot.gov/v2i/
- https://www.its.dot.gov/communications/media/15cv_future.htm
- https://www.iso.org/committee/54706/x/catalogue/
- https://www.iso.org/standard/69897.html
case class Data(key: String, values: Set[String]) extends Iterator[Tuple2[String, String]] {
    private val i = values.iterator
    def hasNext = i.hasNext
    def next = Tuple2[String, String](key, i.next)
}

val d = Seq[((String, Set[String]))](("a", Set[String]("a-1", "a-2", "a-3")))

scala> d.flatMap(i => Data(i._1, i._2))
res3: Seq[((String, String))] = List((a,a-1), (a,a-2), (a,a-3))
Engineering