CVII Tech Stack Workshop:
VSS Data Stores and Servers

Stephen Lawrence, Renesas Electronics
CVII Tech Stack Lead

COVESA
Accelerating the future of connected vehicles

ALL MEMBER MEETING
APRIL 26-28, 2022

What?

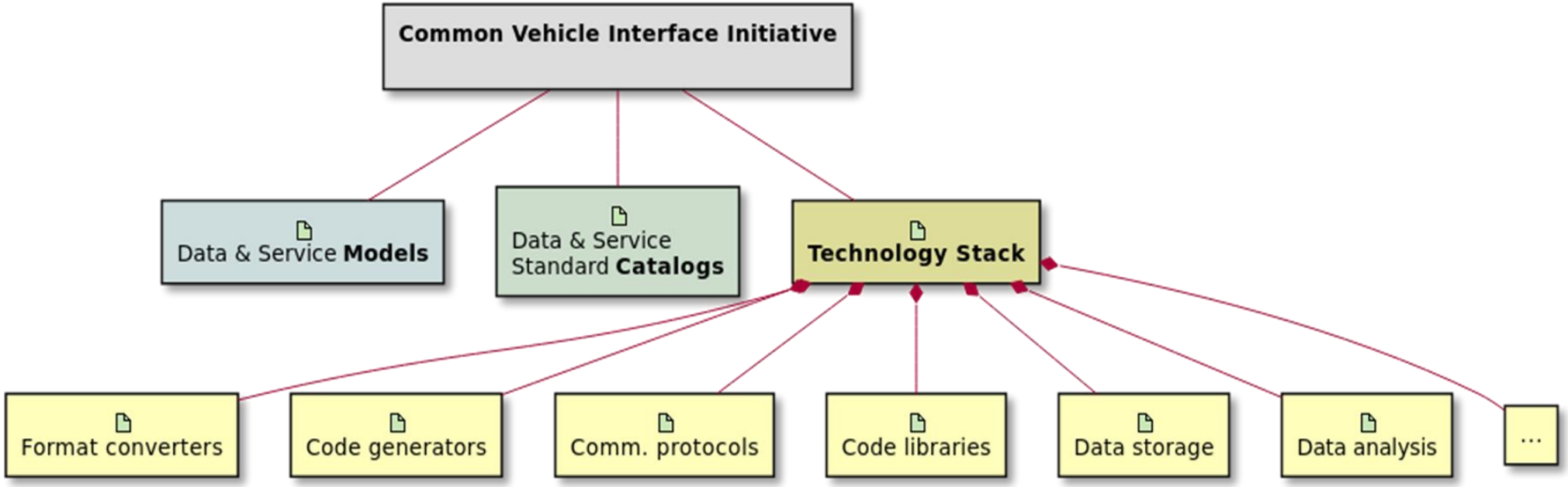
- Quick CVII Tech Stack overview for newcomers
- Workshop
  - VSS Data Storage (including VSS Data Feeders) (Southbound)
    - Feeders: CAN, some/ip
    - Timeseries: return of experience with Apache IoTDB, timeseries vs last value/state storage
    - Interfacing into data servers
  - VSS Data Servers (Northbound)
    - Are VISS backend abstractions possible: storage, feeders?
    - GraphQL: in-vehicle?
- Alts/time allowing
  - Data Store architecture: Single central vs distributed vs direct sensor connect
  - Data Storage news: Redis, timeseries
  - VSS Data Servers to cloud
  - Data processing

CVII Tech Stack Overview

Scope, terminology, architecture diagrams

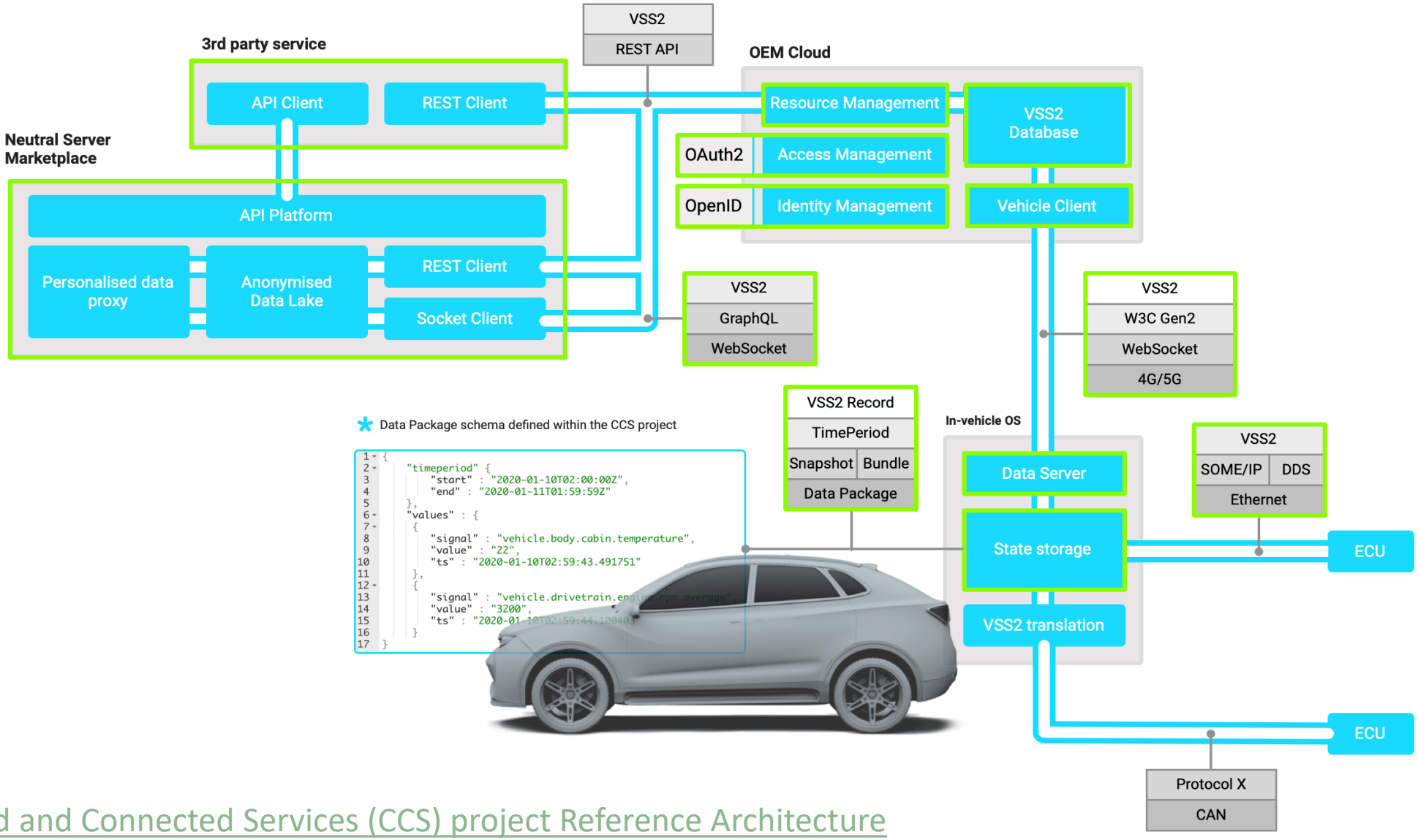
COVESA
Accelerating the future of connected vehicles

Tech Stack Scope



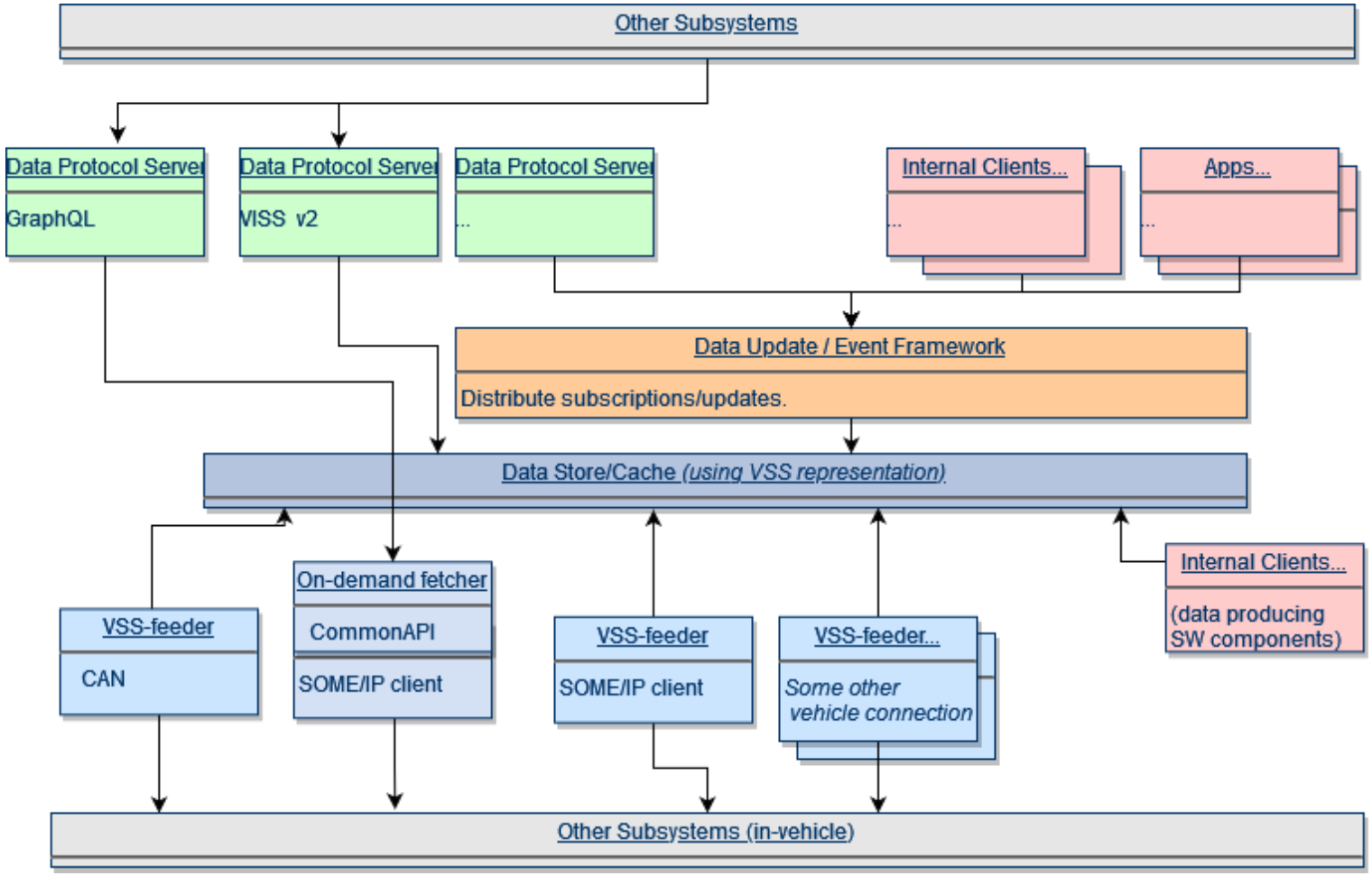
- Technology Stack definition
  - "Any technologies (software) involved in the transfer and use of the *standard data model* and *standard services description model*"
- Defined by:
  - Implementation (Community development projects)
  - Specifications

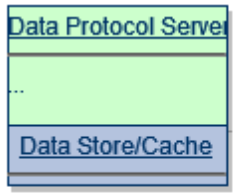
Tech Stack Cloud Architecture*



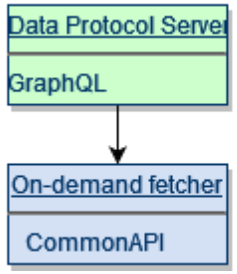
* Cloud and Connected Services (CCS) project Reference Architecture

Tech Stack In-vehicle Architecture*





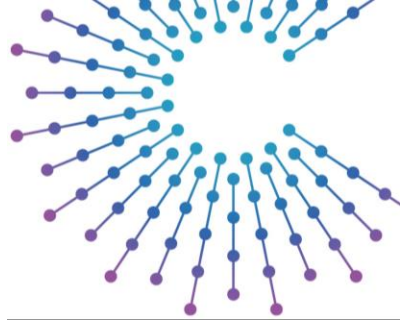
- Some Data servers are closely attached (or include) a data store



- Some Data servers implement connection directly to their data source (skipping a specific data-store component)
- Some fetch data on-demand, when the request comes in and/or through subscription.

* CVII Tech Stack wiki

Tech Stack Project examples



- Specifications, such as W3C VISS Protocol specification
  - Implementations, e.g. servers/clients following W3C spec
- VSS-Tools
  - Collection of tools for conversion from/to VSS
  - Code generation tools (more to do)
  - Franca, JSON, Protobuf, GraphQL, support code for C programming API & Go programming API
  - Android Automotive Vehicle API (Vehicle Properties) from VSS data server (code generation)
- VSC-tools
  - Early implementation of service to code generation
  - Flexible, template driven
- Framework / larger combination projects
  - Aos project
  - KUKSA project
  - PoCs, demos, many internally/proprietary or under development
- Company-internal tools, for VSS, Franca, not open source
- CCS architecture implementation

COVESA
Accelerating the future of connected vehicles

VSS Data Store

VSS Data Feeders

COVESA
Accelerating the future of connected vehicles

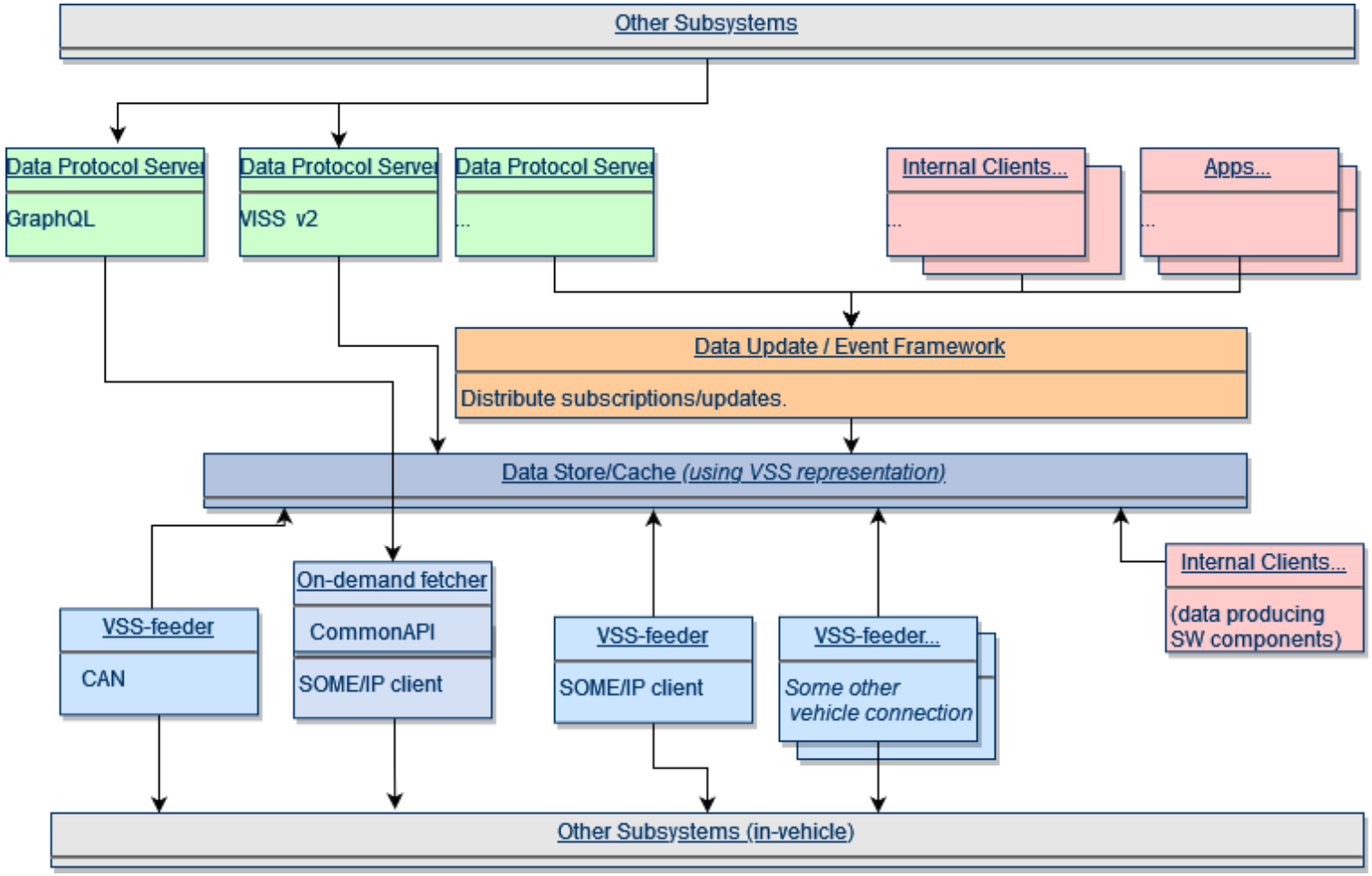
VSS Data Feeder discussion

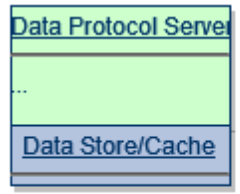
- CAN:
  - CANOPI
  - What are the gaps to fill between CAN and Data Stores?
    - Goal: Create task backlog for people to work on
- Some/ip
  - VSC project proposes to make a connection between Some/IP and VSS/VSC.
  - Requirements to make that a useful stand-alone component?
- Other?

VSS Data Store

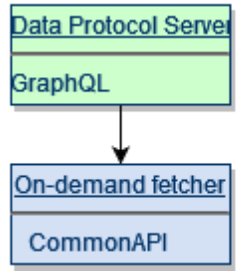
Timeseries (TS) data and databases

Tech Stack In-vehicle Architecture





- Some Data servers are closely attached (or include) a data store



- Some Data servers implement connection directly to their data source (skipping a specific data-store component)
- Some fetch data on-demand, when the request comes in and/or through subscription.

Return of experience: Apache IoTDB

- Renesas contribution to CVII Tech Stack
- Some areas of investigation
  - Introduce production capable timeseries DB to VSS Data Store and Data Server
  - Investigate impact on surrounding components
    - VSS Data Servers: storage, query etc.
    - Feeders, in-vehicle compute etc
- Status
- Next Steps
  - Your ideas?

Needs

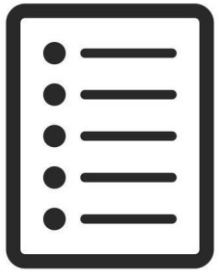
Not exhaustive:

- Handle large data volumes
- High throughput
- Minimise storage footprint
- Support both thin and thick ECUs
- Support in-vehicle/cloud hybrid use cases
  - Survive network connection gaps
  - Aggregation to reduce data vol to cloud
  - Support off-line intelligence

Apache IoTDB

- Why?
  - Designed for Industrial IoT loads, with Edge as first class citizen
  - Meets many of the needs
  - Interesting candidate and general concepts to investigate will apply to other TS DBs..
- New to many so a quick overview..
- Started 2015 in Tsinghua University. Entered incubation in Apache 2018, graduated 2020.
- Already in production use at scale
  - Shanghai metro monitoring, Power plants

Apache IoTDB Features

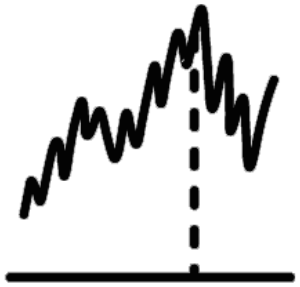


Persist data efficiently

- Millions points ingestion per sec per node
- Tens of millions of time series

Query data with low latency

- Efficiently filter data: millions of points per sec
- Aggregation: tens of ms latency on billions of points

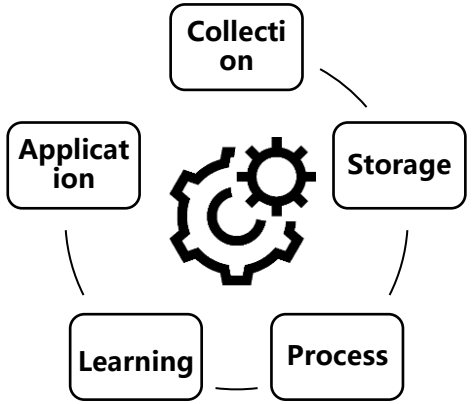


Exclusive operations of time series

- Segmentation
- Representation
- Subsequence matching
- Time-frequency transform
- Visualization

Integration with existing ecosystem

- Kafka
- MatLab
- Spark
- MapReduce
- Grafana

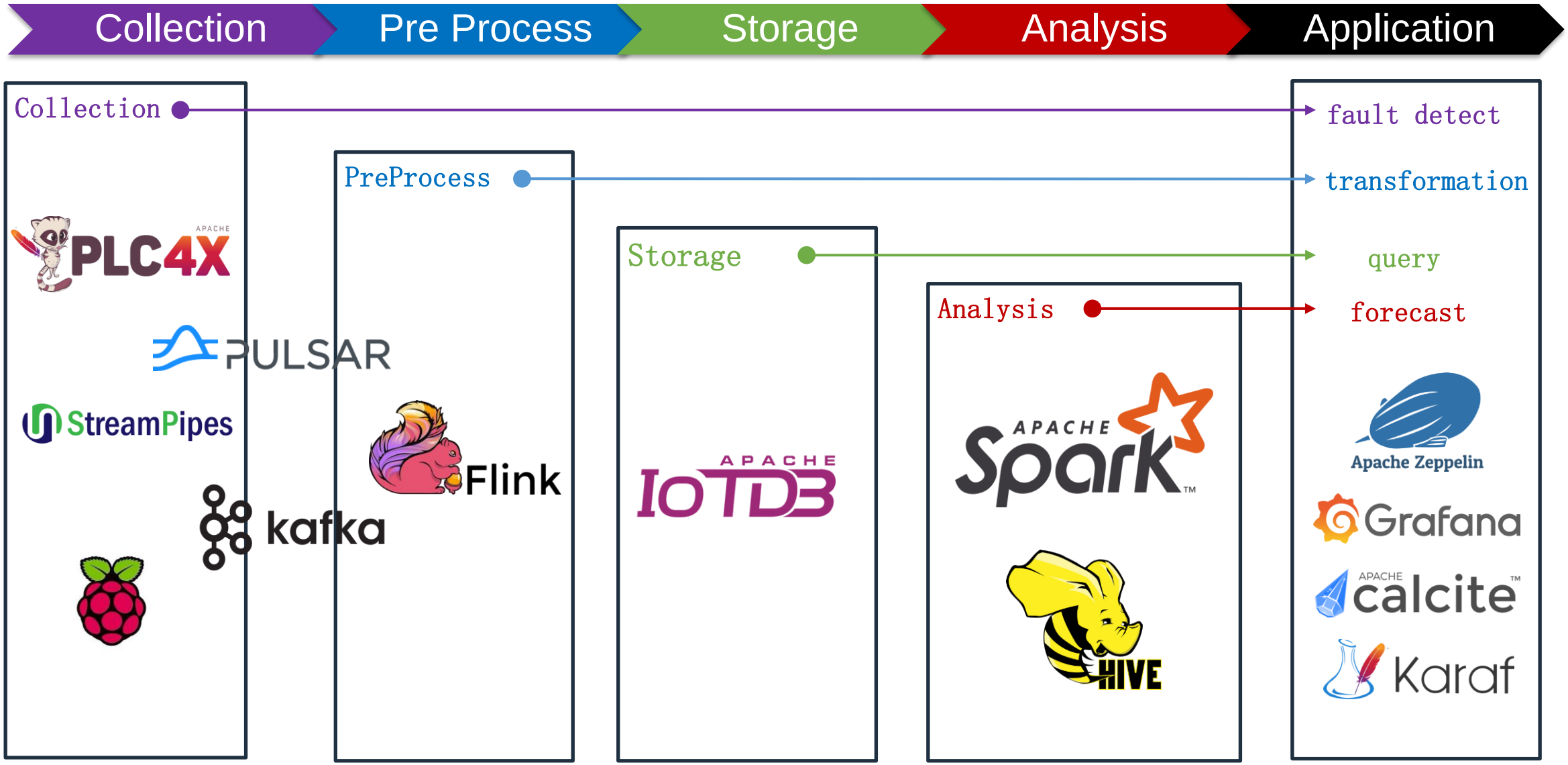


Cover the life cycle of data

- Connecting Edge to the Cloud
- Powerful query engine
- User Friendly analytics

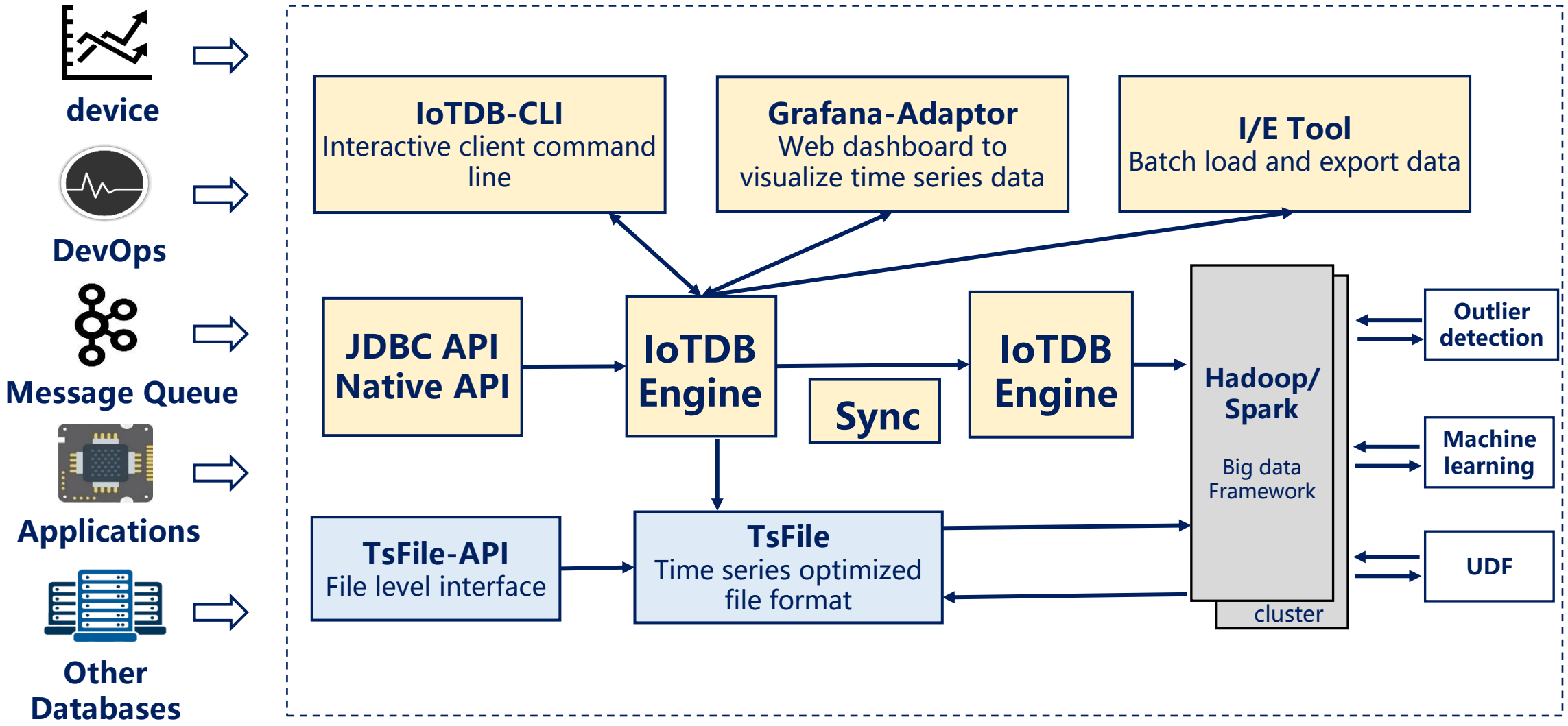
Source: "Apache IoTDB: Time Series Database for Industrial IoT.", Xiangdong Huang, Julian Feinauer. Video

Life cycle of IoT data management



Source: Use Cases and Optimizations of IoTDB, Jialin Qiao.

Architecture of IoTDB System



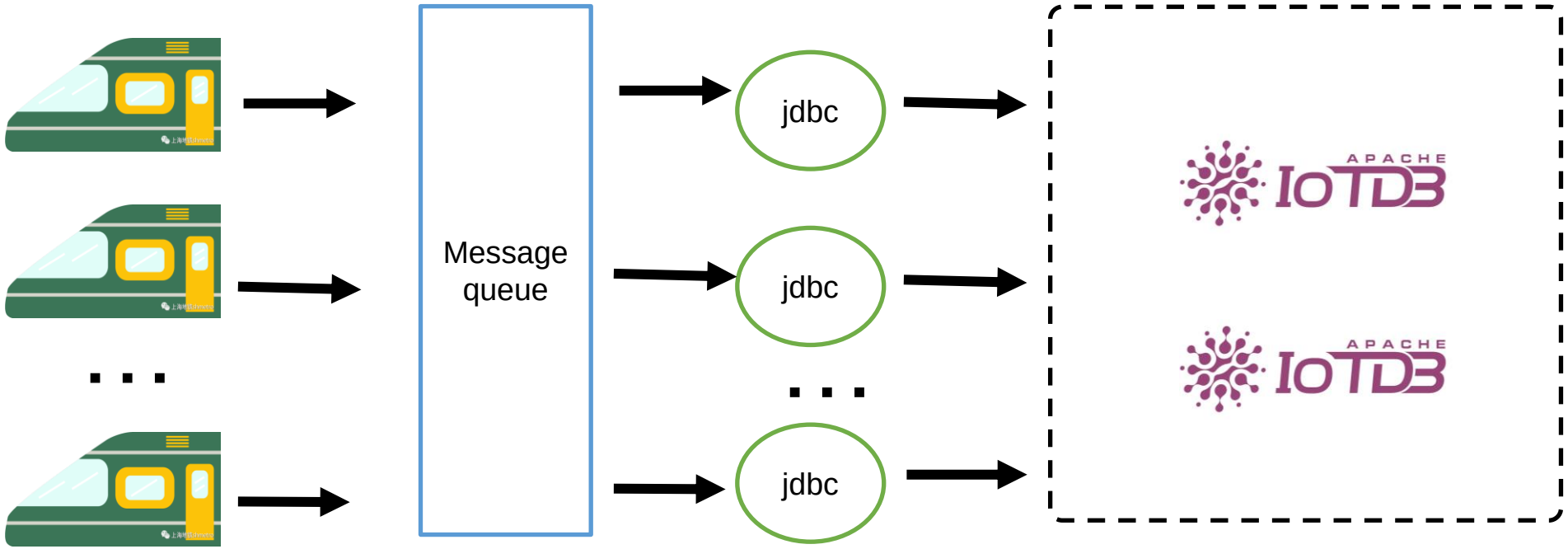
Source: Use Cases and Optimizations of IoTDB, Jialin Qiao.

Subway monitoring application

- 1M time series: 300 subway trains (device) * 3200 sensors in each train
- Frequency: 5Hz, record

Time	Device	s1	...	s3200

- 414 billion points/day, 1TB disk/month



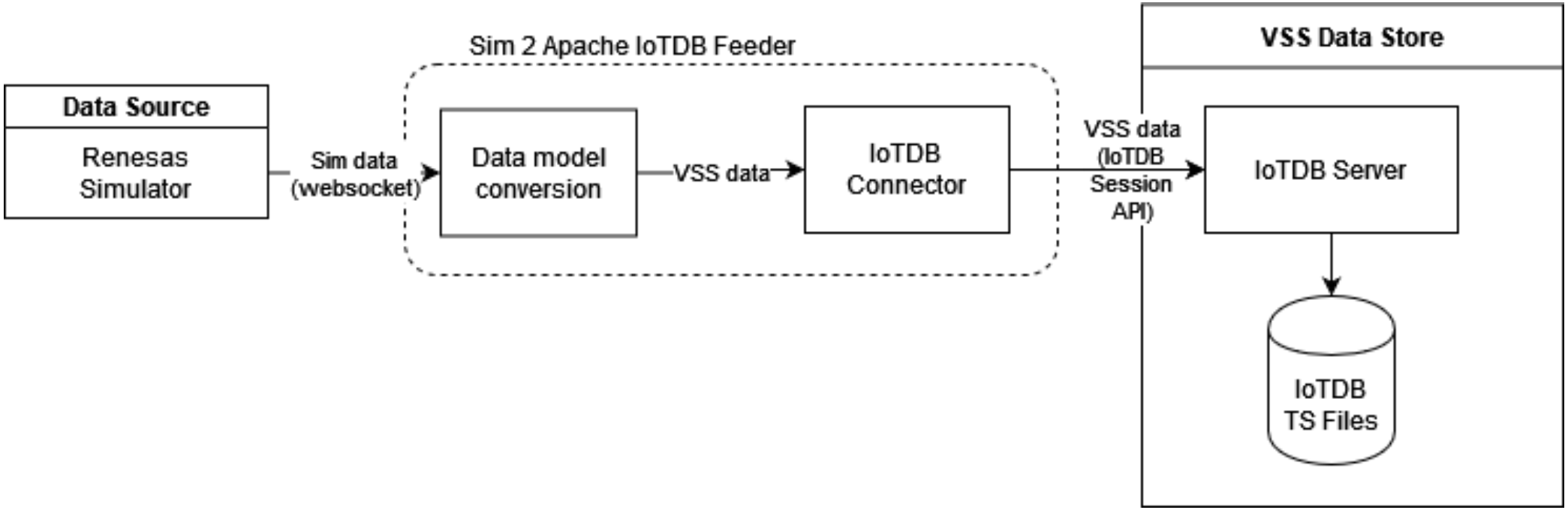
Source: Use Cases and Optimizations of IoTDB, Jialin Qiao.

Status

- Hacking taking place in vss-otaku project on github
  - Design discussion documented for VSS feeders and VSS Data Store
- PoC in Python connecting Renesas Simulator to IoTDB
  - Simple early PoCs to better understand needs and build better
  - Simulator is the one shown at the Showcase last night
  - Production feeder needing high throughput would typically make a direct conversion to maximise speed. Here data model conversion and DB connector are separated.
    - Conversion to Python dictionary creates flexible module for other uses
    - Separating DB connector makes implementation for other DBs simpler

Sim 2 Apache IoTDB Feeder architecture

- Sim writes to websocket so VSS Feeder implemented as websocket server



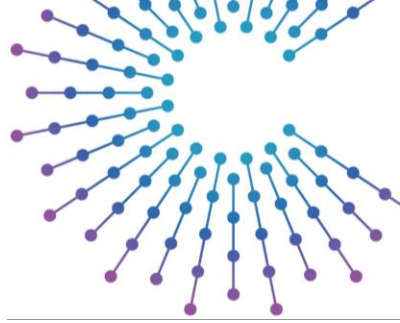
VSS timeseries example

```
IoTDB> show timeseries
+-----------------------------------------------------------+-----+---------------+--------+--------+-----------+----+----------+
|                                                 timeseries|alias|  storage group|dataType|encoding|compression|tags|attributes|
+-----------------------------------------------------------+-----+---------------+--------+--------+-----------+----+----------+
|root.sg_test_01.d_01."Vehicle.CurrentLocation.Longitude"| null|root.sg_test_01|   FLOAT|   PLAIN|     SNAPPY|null|      null|
| root.sg_test_01.d_01."Vehicle.CurrentLocation.Latitude"| null|root.sg_test_01|   FLOAT|   PLAIN|     SNAPPY|null|      null|
+-----------------------------------------------------------+-----+---------------+--------+--------+-----------+----+----------+
Total line number = 2
It costs 0.016s


IoTDB> select * from root.*
+-----------------------+-----------------------------------------------------------+-----------------------------------------------------------+
|                   Time|root.sg_test_01.d_01."Vehicle.CurrentLocation.Longitude"|root.sg_test_01.d_01."Vehicle.CurrentLocation.Latitude"|
+-----------------------+-----------------------------------------------------------+-----------------------------------------------------------+
|2022-03-24T20:12:46.353Z|                                                  -73.98506|                                                   40.76432|
|2022-03-24T20:12:47.404Z|                                                  -73.98519|                                                   40.76438|
|2022-03-24T20:12:48.419Z|                                                  -73.98531|                                                   40.76443|

IoTDB> select last_VALUE("Vehicle.CurrentLocation.Longitude") from root.sg_test_01.d_01
+-------------------------------------------------------------------+
|last_VALUE(root.sg_test_01.d_01."Vehicle.CurrentLocation.Longitude")|
+-------------------------------------------------------------------+
|                                                           -73.99346|
+-------------------------------------------------------------------+
Total line number = 1
It costs 0.006s
```

Next Steps



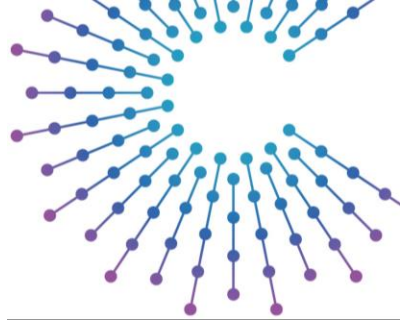
- Near term planned
  - Upgrade to IoTDB v0.13
    - For aligned record inserts and expanded data quality functions
  - Connection to VISS Data Servers
    - WAII: Connection to CCS State Storage component.
    - Kuksa.val: investigation of storage abstraction needed

- Mid-term under consideration
  - Your ideas?
  - VSS vspec -> IoTDB schema generation
  - Investigate impact of TS on current VSS Data Server protocols, e.g. TS query
  - Automotive h/w
  - PoC connection to GraphQL Resolver
  - Data quality
  - Investigate VSS data model in IoTDB eco-system

COVESA
Accelerating the future of connected vehicles

VSS Data Servers

VISS backend abstractions



- Abstractions for VSS Data Store and Data Feeders would be beneficial
  - Write once, deploy many, e.g. to WAII and kuksa.val
- Possible? If yes, what is the path?
- WAII example:
  - Uses CCS State Storage component (SSC) as both feeder and data store backend
    - WAII supports SQLite or Redis as “last value” only VSS Data Store
    - SSC provides get/set data abstraction APIs, but for single key/value pair
    - This is the abstraction for both Feeder southbound and Server northbound.
- Kuksa.val?

COVESA
Accelerating the future of connected vehicles

GraphQL

- Anything you want to discuss?
- Anyone considering its use in-vehicle?

In-vehicle zonal architecture

Coordination between VSS Data Stores and or Servers

Example for discussion

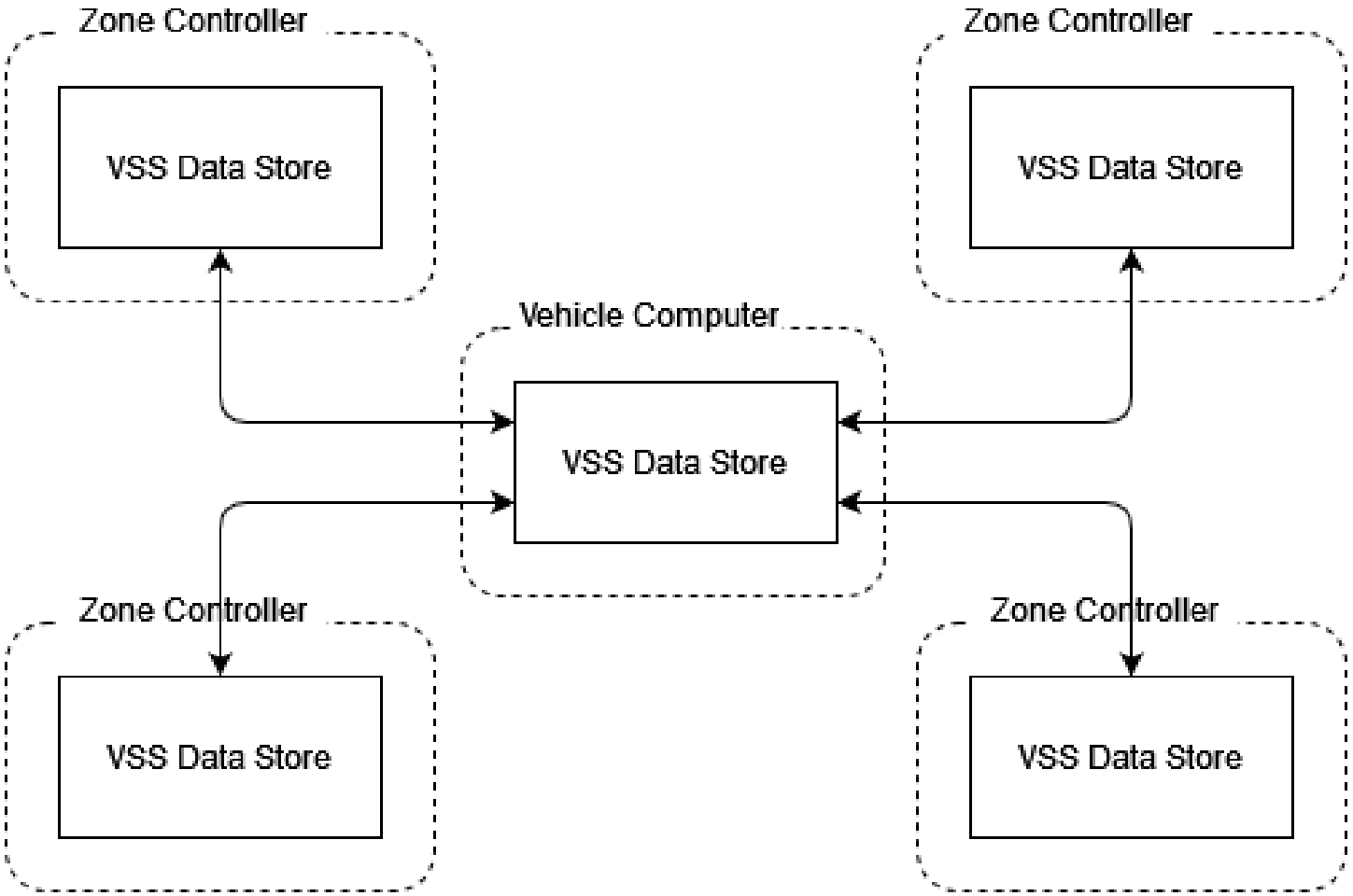


Figure: Zone Controllers performing data cooperation with central Vehicle Computer VSS Data Store

- Multiple Data Stores?
  - Sync method? Native or open?
- Vehicle Computer provides Server?
- Obviously implementation specific but what is common? Missing?

COVESA

Accelerating the future of connected vehicles