Design and Development of a Synchronized Database for Launch Control System

Tatia Tibunashvili
Advisor: Julian Petrasch
Chair of Distributed Systems and Operating Systems
https://dse.in.tum.de/

15.11.2022 – 15.05.2023

Outline

- Motivation
- Industry-wide approach
- Problem Statement
- System selection
    - Database selection
    - Comprehensive assessment of the chosen database
    - Introducing streaming service into the system
- Conclusion and final system layout

Motivation: Reliable data glow and real-time monitoring

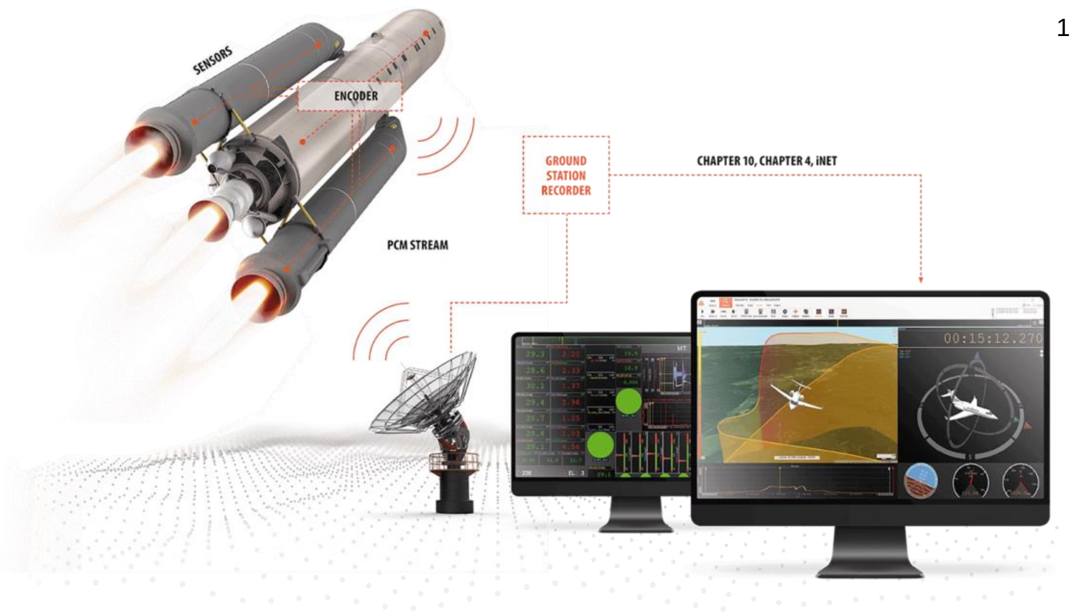
- A reliable data flow and governance are essential for mission monitoring systems
- The ground system enables engineers to monitor spacecraft health, gather valuable post-launch data, and analyze the system's performance

[1]

[1] isaraerospace.com

Industry-wide approach

- SLS (Space Launch System - NASA) is currently using products provided by Dewesoft
- eZprocessing from from Safran DataSystems
- Limited information about the the industry standards
- General approach: telemetry data acquisition/storage models are offered as a package of hardware and software



[1]

[1] https://dewesoft.com/products/dewesoftx

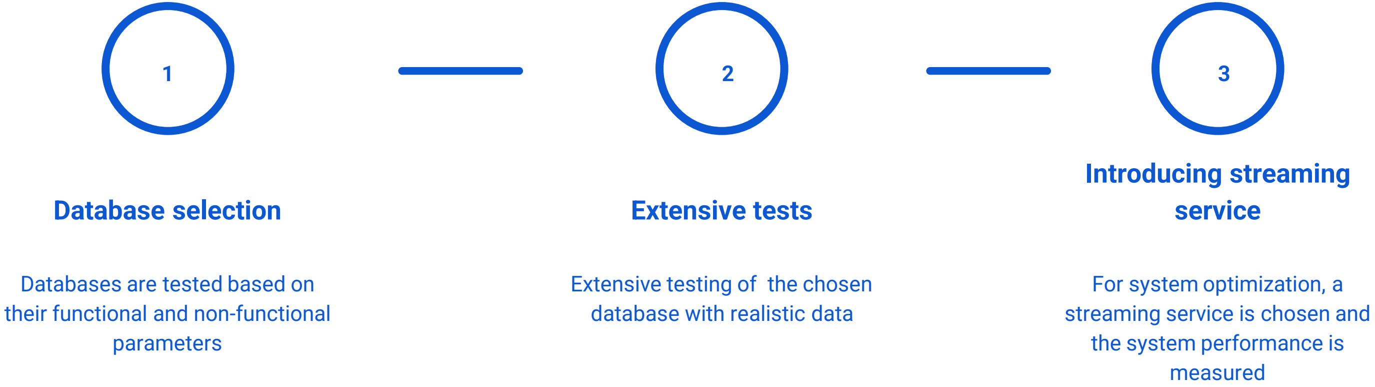
Problem statement

- Balancing the need for persistent data storage and real-time availability is a challenge
- Data should be filtered and divided among multiple streams to cater to the specific needs of different engineers.
- System must perform even under high data load

[1]

[1] https://shorturl.at/dijxZ

The sequence of tests



1

Database selection

Databases are tested based on their functional and non-functional parameters

2

Extensive tests

Extensive testing of the chosen database with realistic data

3

Introducing streaming service

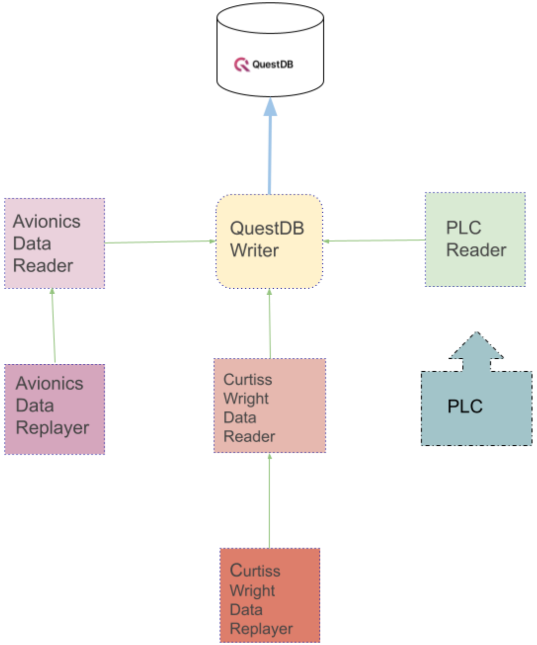
For system optimization, a streaming service is chosen and the system performance is measured

Database selection

- Focus on time-series databases
- Uniform environment for the initial tests
- 5 different databases selected:
    - AerospikeDB
    - QuestDB
    - TimescaleDB
    - Druid
    - InfluxDB

Comprehensive assessment of QuestDB

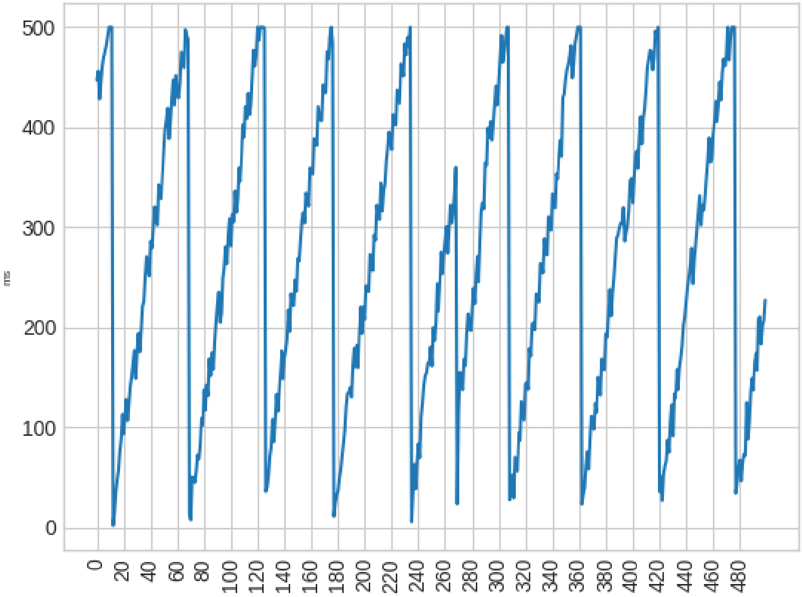
- Second phase tests introduced additional data streams
  - Flight test instrumentation system (50 Hz)
  - PLCs (10 Hz and 1 kHz)
  - Avionics stream from the launch vehicle (25 Hz)
- Two types of tests conducted
  - Server writes without querying
  - full system performance



Comprehensive assessment of QuestDB

- Garbage collection had an impact on timing measurements
- As the polling frequency increases, the proportion of requests yielding new results decreases, resulting in increased database overhead
  - Intended period: 200 ms
  - Actual period: 500 ms
- Every hour about 5% of reads fail (query returns no data)

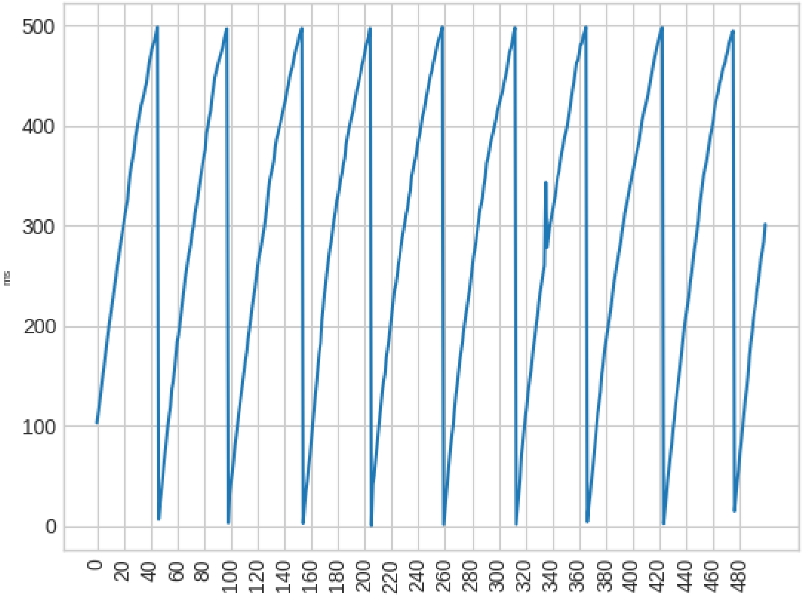
Time difference between query initialization and the most recent timestamp read (avionics data)



Comprehensive assessment of QuestDB

- PLC frequency range
  - 1 Hz to 1 kHz
- Selected frequencies:
  - Most of data at 10 Hz
  - Selected sensors at 1 kHz
- Implemented via pyads
- Bottleneck: queue shared between processes
- Conclusion:
  - Writing can handle the load
  - Reading needs an improvement

Time difference between query initialization and the most recent timestamp read (InetX stream)



Introducing streaming service into the system

- Apache Kafka: Industry standard for streaming platforms
- Durability is crucial for data processing before reaching the database
- Kafka features:
  - better throughput
  - built-in partitioning and replication

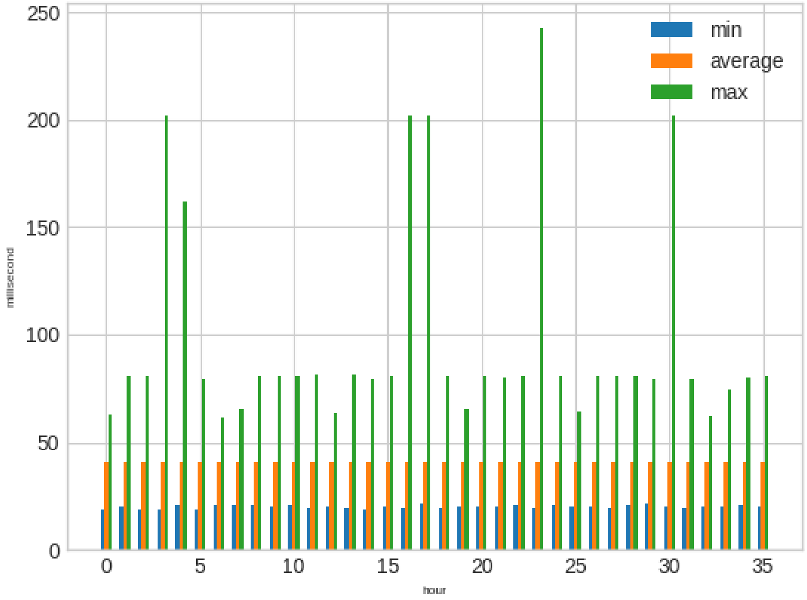
[1]

[1] https://commons.wikimedia.org/wiki/File:Apache_kafka_wordtype.svg

Introducing streaming service into the system

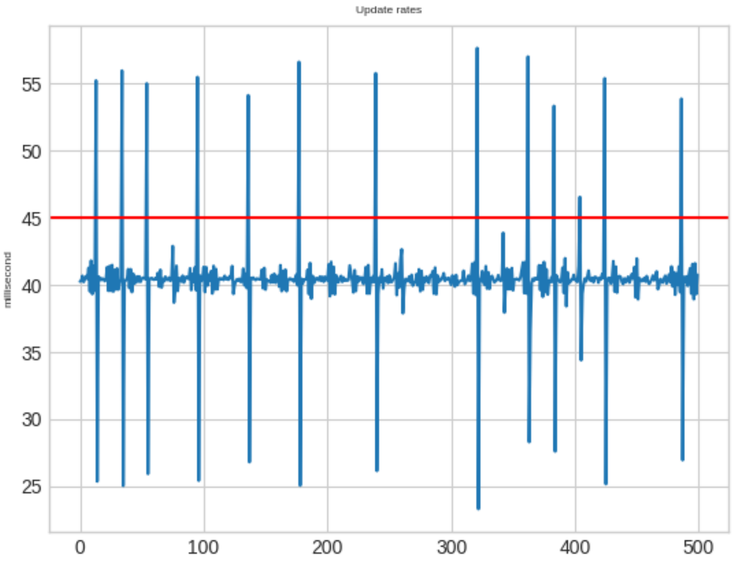
- Kafka producer's send() is 10 times slower than appropriate QuestDB ingestion function
- Makes data available almost at the rate of transmission.
- Completely different way of fetching data compared to the database

Avionics data read pattern

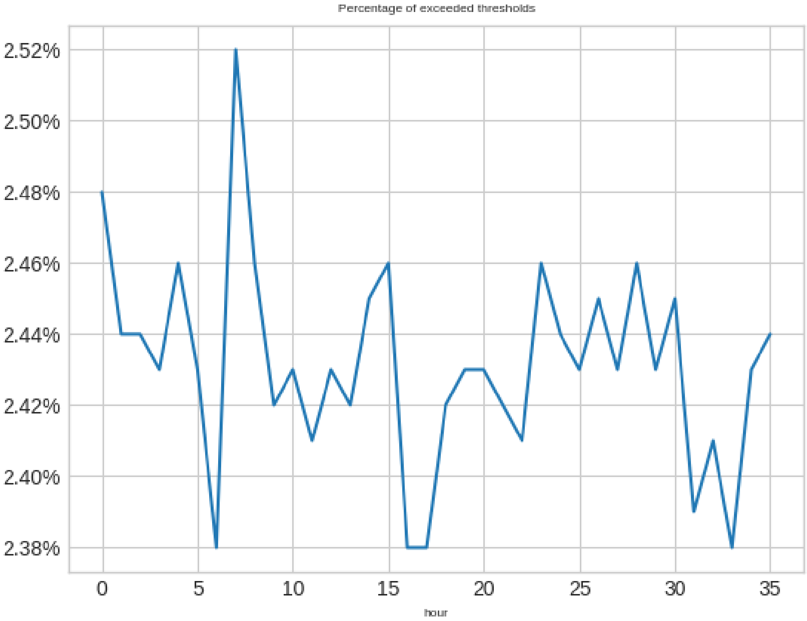


Introducing streaming service into the system

Reading pattern for avionics data



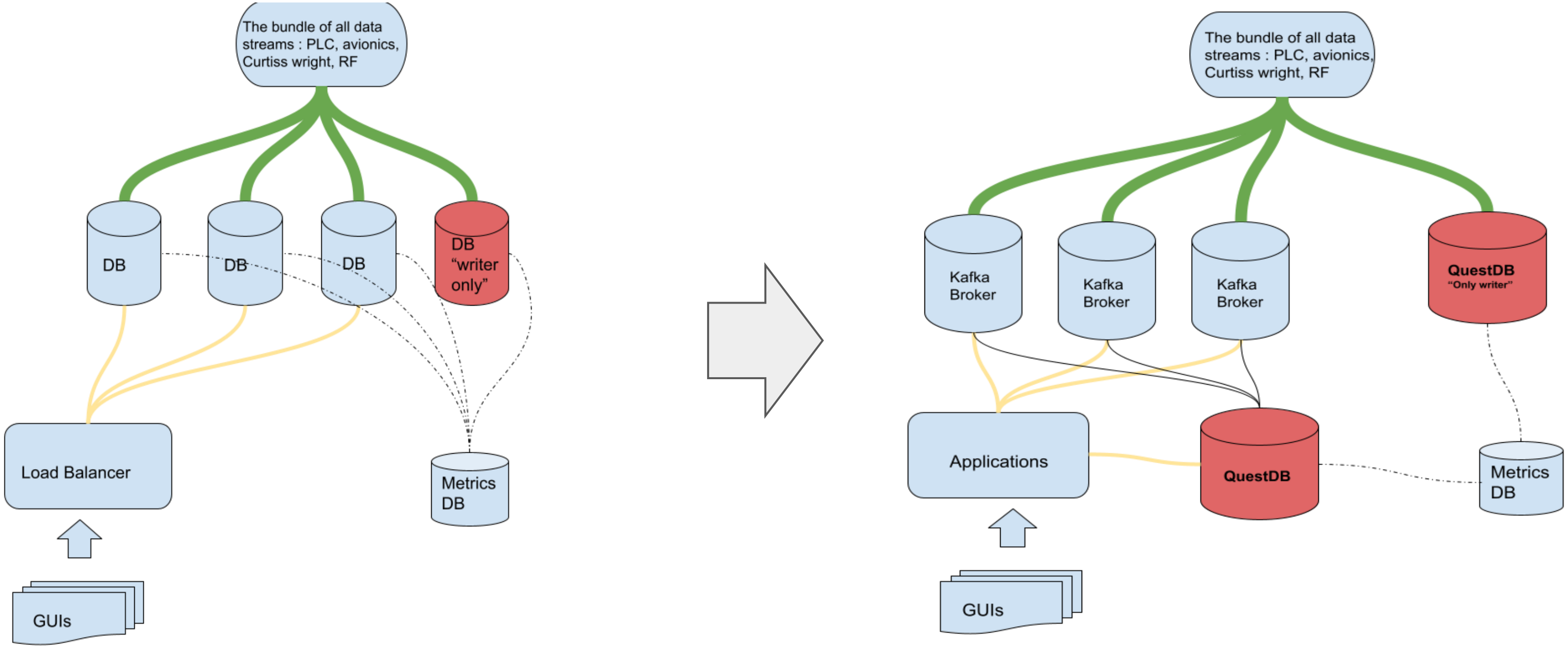
Percentage of exceeded threshold (45 ms) per hour (avionics data)



Conclusion and the final system layout

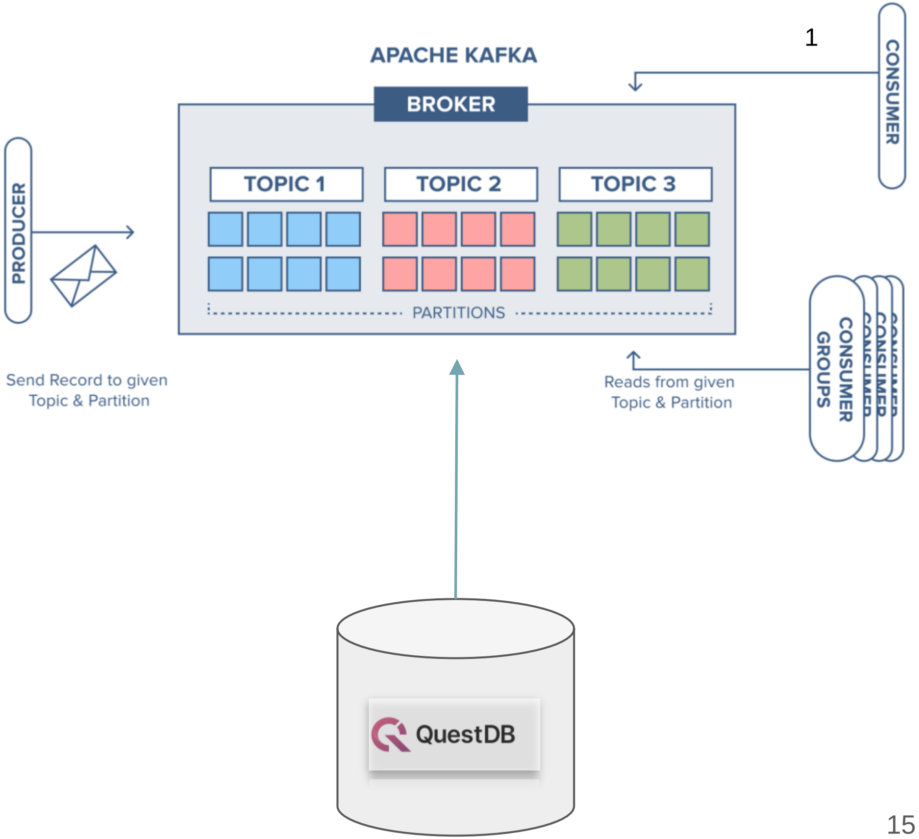
Initial layout

Final layout



Conclusion and the final system layout

- Inspiration coming from Druid
- The streaming service for streaming data
- The database for “historical” data
- The streaming service allows multiple applications to subscribe to different Kafka topics
- The prototype of a central database for the rocket launch was built as a result of the thesis



[1] https://www.cloudkarafka.com/blog/part1-kafka-for-beginners-what-is-apache-kafka.html