Kpi engine : State Of The Art Analysis

This is the analysis done to narrow done possible technologies to use, some choices are influenced by the fact that FEAST was choosen as the main database to use.

Requirements and Constraints

To handle real-time data processing effectively, we need to evaluate technologies based on specific requirements and constraints:

1. **Constraints**:

    - **Disk Usage**: Minimize the amount of data written to disk, especially if working with high-throughput streams.
    - **Memory Usage**: Balance memory consumption, given the demands of real-time data processing.

2. **Needs**:

    - **Fast Calculations**: Support for quick computations is crucial for real-time responses.
    - **Complex Queries**: Ability to perform sophisticated analyses and aggregations.
    - **High Throughput**: Sustainably process large volumes of incoming data.

Evaluated Technologies

1. **PostgreSQL/TimescaleDB**

- **Description**: An extension of PostgreSQL tailored for time-series data, enabling efficient storage and querying of time-based information.

- **Pros**:

    - **Compatibility with FEAST**: Supports integrations with FEAST.
    - **Efficient Data Retrieval**: Optimized for high-performance data retrieval, even with large volumes of time-series data.
    - **Rich query capabilities**: It offers very complex and complete query capabilities useful for mixing various data types.
    - **Good stability and transaction integrity**:It offers good stability and transaction integrity making it very reliable.

- **Cons**:

    - **Potentially Higher Storage Costs**: Disk usage may increase depending on the frequency of data collection and retention policies.
    - **Setup Complexity**: Configuration and optimization for high-throughput applications can be difficult.

2. **Redis**

- **Description**: An in-memory, multi-paradigm datastore designed for speed and low latency, often used as a caching layer or real-time processing component.

- **Pros**:

    - **Very Fast, Low Latency**: In-memory processing enables rapid calculations, making Redis highly responsive.
    - **Scalability**: Easy to scale horizontally, can perform well in low latency scenarios.
    - **Persistence Options**: Can be configured for in-memory only, or with persistence options.

- **Cons**:

    - **Memory Constraints**: All data is stored in memory, limiting the size of datasets that can be handled without significant memory resources.
    - **Limited Querying Capabilities**: Lacks advanced querying structures, restricting complex queries.
    - **Challenges with Large Datasets**: Handling very large datasets can become costly or impractical due to memory requirements.

3. **InfluxDB**

- **Description**: A purpose-built time-series database optimized for high-throughput and low-latency scenarios.

- **Pros**:

    - **High Throughput and Low Latency**: Optimized for continuous data ingestion and real-time access.
    - **Time-Series Optimization**: Native support for time-series data structures, allowing efficient storage and retrieval.
    - **Continuous Queries and Downsampling**: Provides built-in support for ongoing calculations and data downsampling to reduce storage needs.

- **Cons**:

    - **Specialized for Time-Series**: While this is a benefit for time-series use cases, it can limit general-purpose usage, in our case it not necessarily a down side.
    - **Limited FEAST Integration**: InfluxDB lacks native integration with FEAST but could potentially be adapted in an easy manner.
    - **Restricted Query Capabilities**: No support for complex queries, it might be a problem in the long run.

- **Suitability**: InfluxDB is highly suitable for real-time data scenarios, especially with time-series data.

Preliminary Conclusion

It is too early to talk about a final choice given the uncertainty of data handling and need for other essential features that have priority. Given the FEAST architecture we might use more of them at the same time.