Visual Exploration of Time Series Anomalies with Metro-Viz

Philipp Eichmann (Brown University); Franco Solleza (Brown University);
Junjay Tan (Brown University); Nesime Tatbul (Intel Labs and MIT); Stan Zdonik (Brown University)

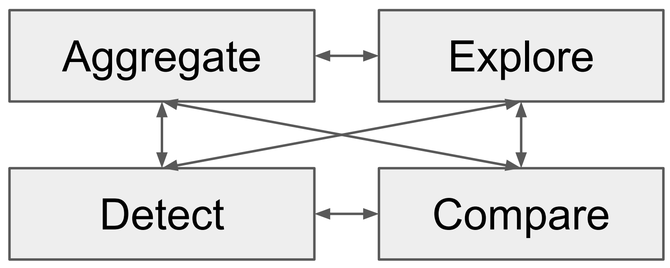
Motivation and Core Principles

The Problem

- **Anomalies** are patterns that do not conform to expected behavior.
- Time Series **Anomaly Detection** is particularly challenging because:
  - Anomalies are **domain and context specific**
  - There is typically little or no available **ground-truth**
- This requires **domain experts to explore and compare** the results of black-box detectors so that they:
  - understand the **characteristics of different detectors** on their data and,
  - are able to better **infer their behavior in hypothetical scenarios** not present in the data

Our Solution

- Metro-Viz helps domain experts **visually analyze time series data and detector performance** through four key features:
  - browsing and inspecting anomalies (regardless of the size of the data)
  - filter anomalies using key properties
  - probe detector behavior through counter-factuals
  - evaluate detectors using interactively built ground truth
- This presents a unique workload to a data management system:

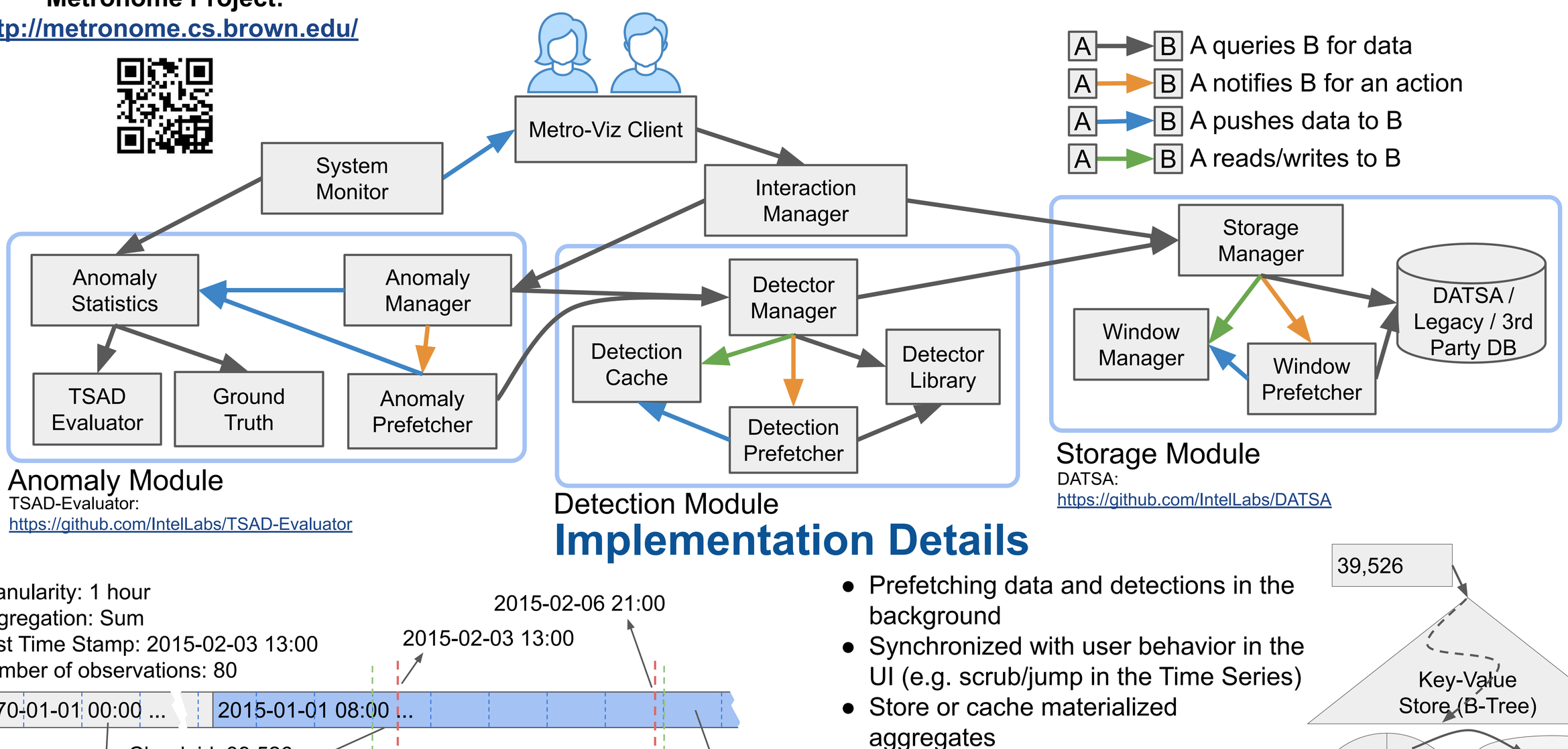


Goals, Constraints, and Approach

- Goals:
  - Maintain **interactive latency** (e.g. 60 fps) or **keep user informed** of system activity
  - Understand **challenges** and **requirements** posed by workload
- Constraints:
  - Cannot assume user access patterns in their data (e.g. **arbitrary granularities / windows**)
  - Data sources come from **legacy/3rd party data systems**
  - Users **should not wait** to see results
  - Cannot rely on client-side data management
- Approach:
  - **UI-DB co-design**: APIs that access system information
  - Architecture **prioritizes user interaction** over background tasks

Metronome Project:
http://metronome.cs.brown.edu/

Architectural Overview

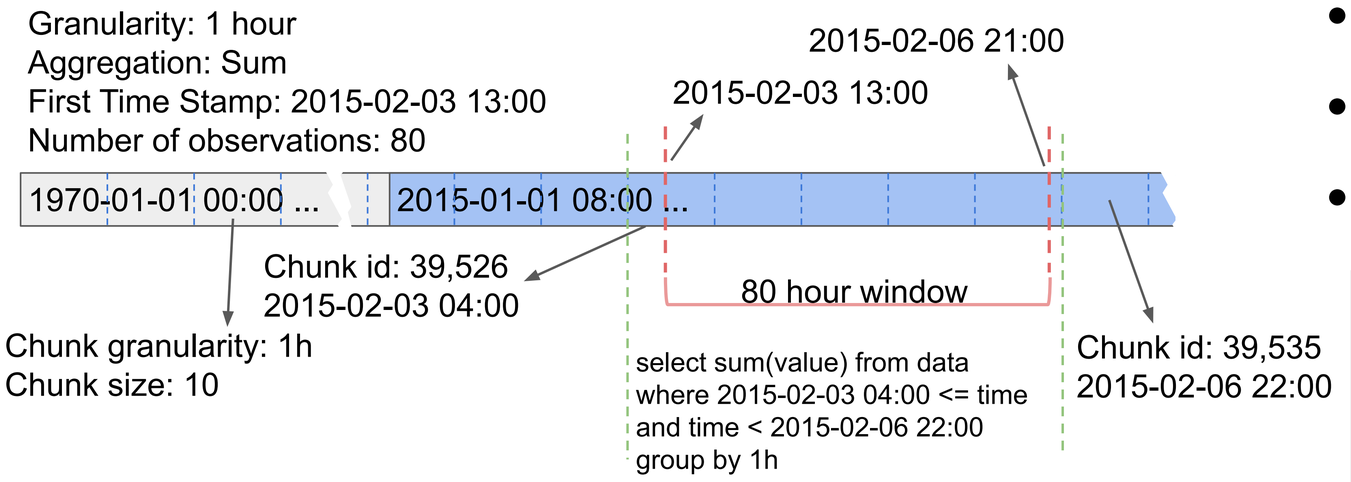


Anomaly Module
TSAD-Evaluator:
https://github.com/IntelLabs/TSAD-Evaluator

Detection Module

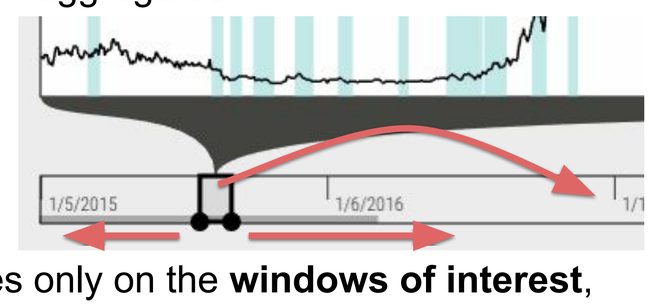
Storage Module
DATSA:
https://github.com/IntelLabs/DATSA

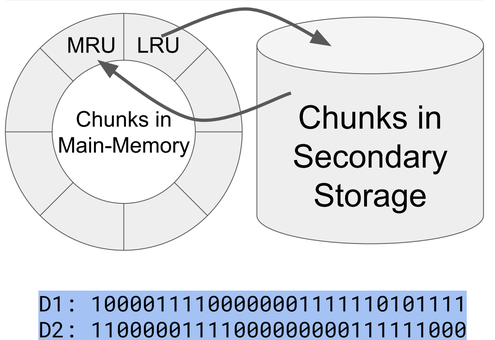
Implementation Details



- **Chunk** time to discrete fixed sized chunks from Unix Epoch
- **Windows** are a collection of these chunks
- Query underlying data storage for chunks in time

- Prefetching data and detections in the background
- Synchronized with user behavior in the UI (e.g. scrub/jump in the Time Series)
- Store or cache materialized aggregates





- Detect anomalies only on the **windows of interest**, background detections on other windows
- Detections are **first-class data citizens**: prefetch and store them as data, not metadata
- Detections stored as **bit-vectors** for efficient set and comparison operations

```
D1: 10000111100000001111110101111
D2: 11000001111000000000111111000
D3: 00001100001111100011111110000

A Only: 11000011110000001100000001111
B Only: 00001000000111100000000000000
A and B: 00000100001000000000111110000
Consensus: 21001202212111101122332322111
```

Take-Aways

- **Interactive latency**: Throughput matters less than latency. Response in the order of < 30ms required - any slower leads to poor interactive interface.
- **Aggregation bottleneck**: Time-series AD requires arbitrary granularity. Most databases assume granularities a-priori.
- **Ratio of chunk-size** and **base granularity**: determines how much data is touched and directly affects latency.
- **Bit-vectors work well** for the many operations involved in comparing anomaly detection results.

Future Work

- Explore **index and cache techniques** to address **aggregation bottleneck** while **considering chunk-size and granularity** to maintain **interactive latency at scale**
- **User study** measuring the efficacy of Metro-Viz UI in assisting in the HiL workload