TMLL

Trace-Server Machine Learning Library

Kaveh Shahedi, Matthew Khouzam

Ericsson

Summer/Fall 2024

What is Trace Compass?

Trace Compass is a tool for solving **performance** and **reliability** issues by **analyzing** system **traces**, providing user-friendly **views**, **graphs**, and **metrics**.

What is a Trace?

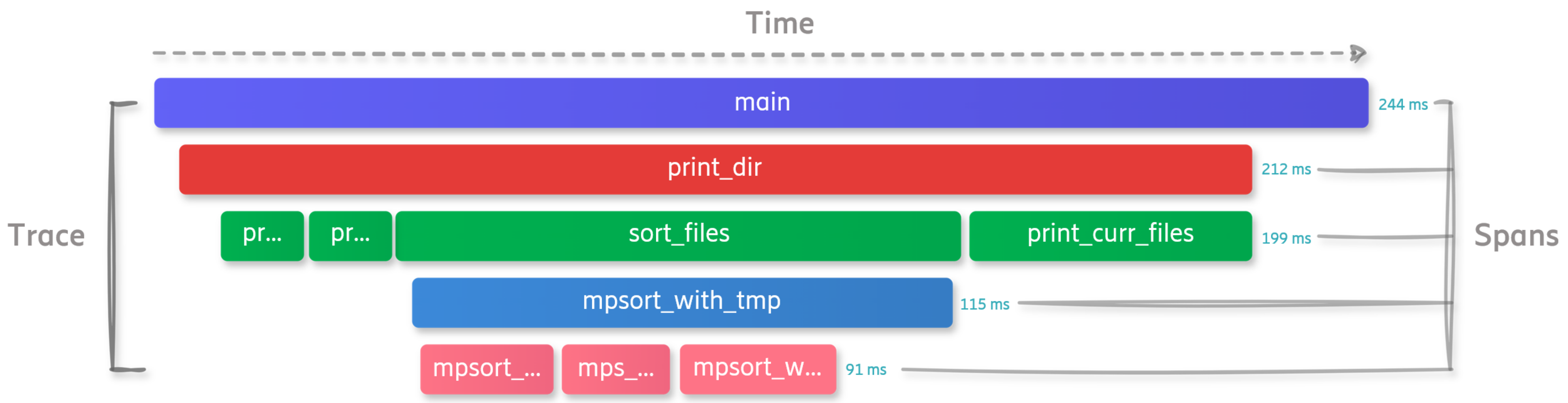
Structure (example)

```
[Timestamp]   [Process ID]   [Thread ID]   [Location]   Additional Data
```

Example

```
2024-10-28T10:00:00Z [PID: 54300] [Thread ID: 1] [main:10] Entering function main
2024-10-28T10:00:01Z [PID: 54300] [Thread ID: 1] [print_dir:20] Entering function print_dir
2024-10-28T10:00:02Z [PID: 54300] [Thread ID: 1] [sort_files:30] Entering function sort_files
2024-10-28T10:00:03Z [PID: 54300] [Thread ID: 1] [sort_files:30] Exiting function sort_files
2024-10-28T10:00:03Z [PID: 54300] [Thread ID: 1] [print_curr_files:40] Entering function print_curr_files
2024-10-28T10:00:04Z [PID: 54300] [Thread ID: 1] [print_curr_files:40] Exiting function print_curr_files
2024-10-28T10:00:04Z [PID: 54300] [Thread ID: 1] [print_dir:20] Exiting function print_dir
2024-10-28T10:00:05Z [PID: 12902] [Thread ID: 2] [mp_sort_with_tmp:50] Entering function mp_sort_with_tmp
2024-10-28T10:00:06Z [PID: 12902] [Thread ID: 3] [mp_sort_inner:60] Entering function mp_sort_inner
2024-10-28T10:00:06Z [PID: 12902] [Thread ID: 3] [mp_sort_inner:60] Exiting function mp_sort_inner
2024-10-28T10:00:07Z [PID: 12902] [Thread ID: 2] [mp_sort_with_tmp:50] Exiting function mp_sort_with_tmp
2024-10-28T10:00:07Z [PID: 54300] [Thread ID: 1] [main:10] Exiting function main
```

Visual


Time
main
244 ms
print_dir
212 ms
pr…
pr…
sort_files
print_curr_files
199 ms
mpsort_with_tmp
115 ms
mpsort_…
mps_…
mpsort_w…
91 ms
Trace
Spans

What is Trace Server?

An **open-source tool** for **analyzing logs** and **traces**, enabling users to interact via **client-side wrappers** in **Python** and **TypeScript**, and providing **various analyses** and **outputs** without requiring direct use of Trace Compass.

What can Trace Server provide?

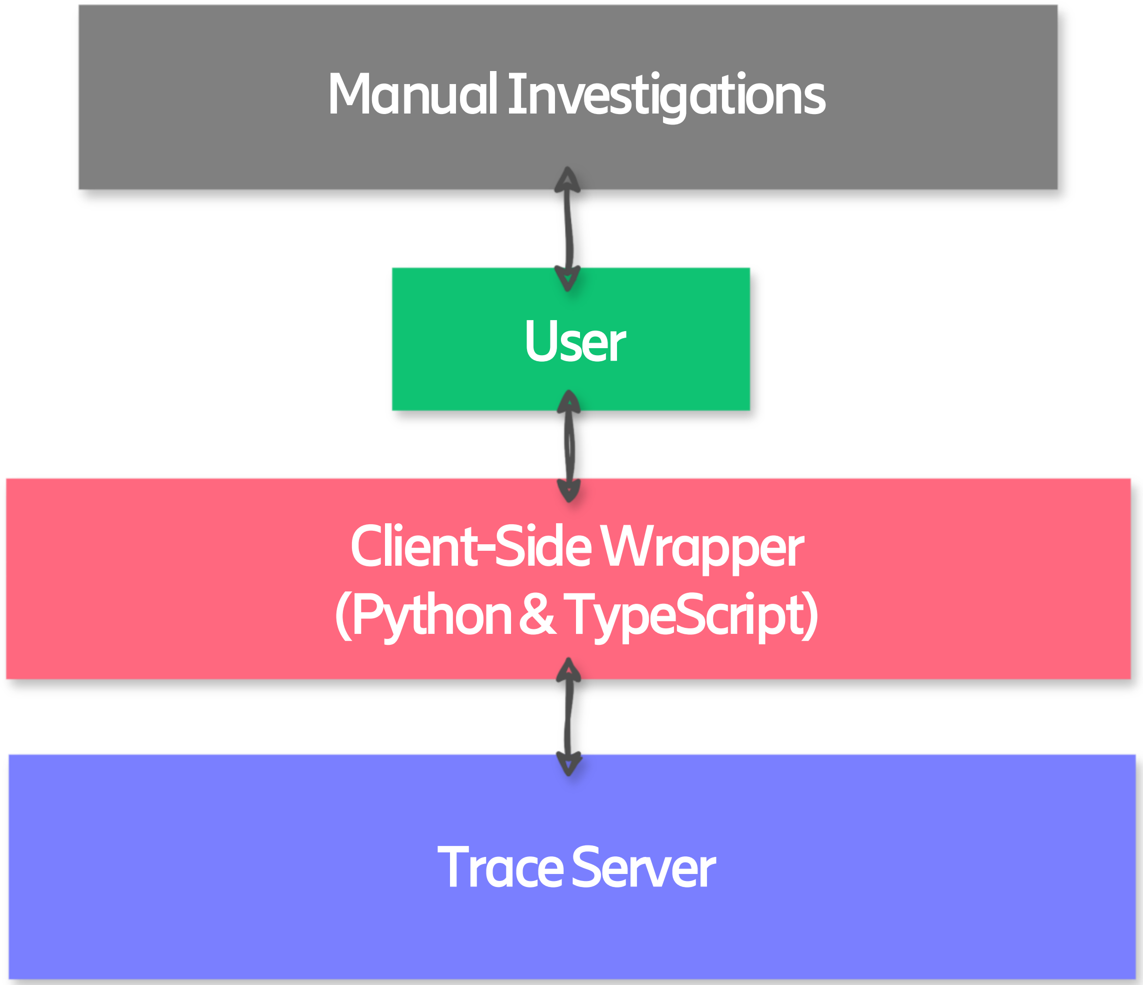
Almost 40 Various analyses based for both user-space and kernel trace data

- CPU and thread performance
- Disk I/O performance
- Memory usage and latency
- Event analysis
- Function call analysis
- Futex contention analysis (i.e., synchronizations)
- IRQ analysis (i.e., interrupts)
- Scheduler latencies
- System call analysis

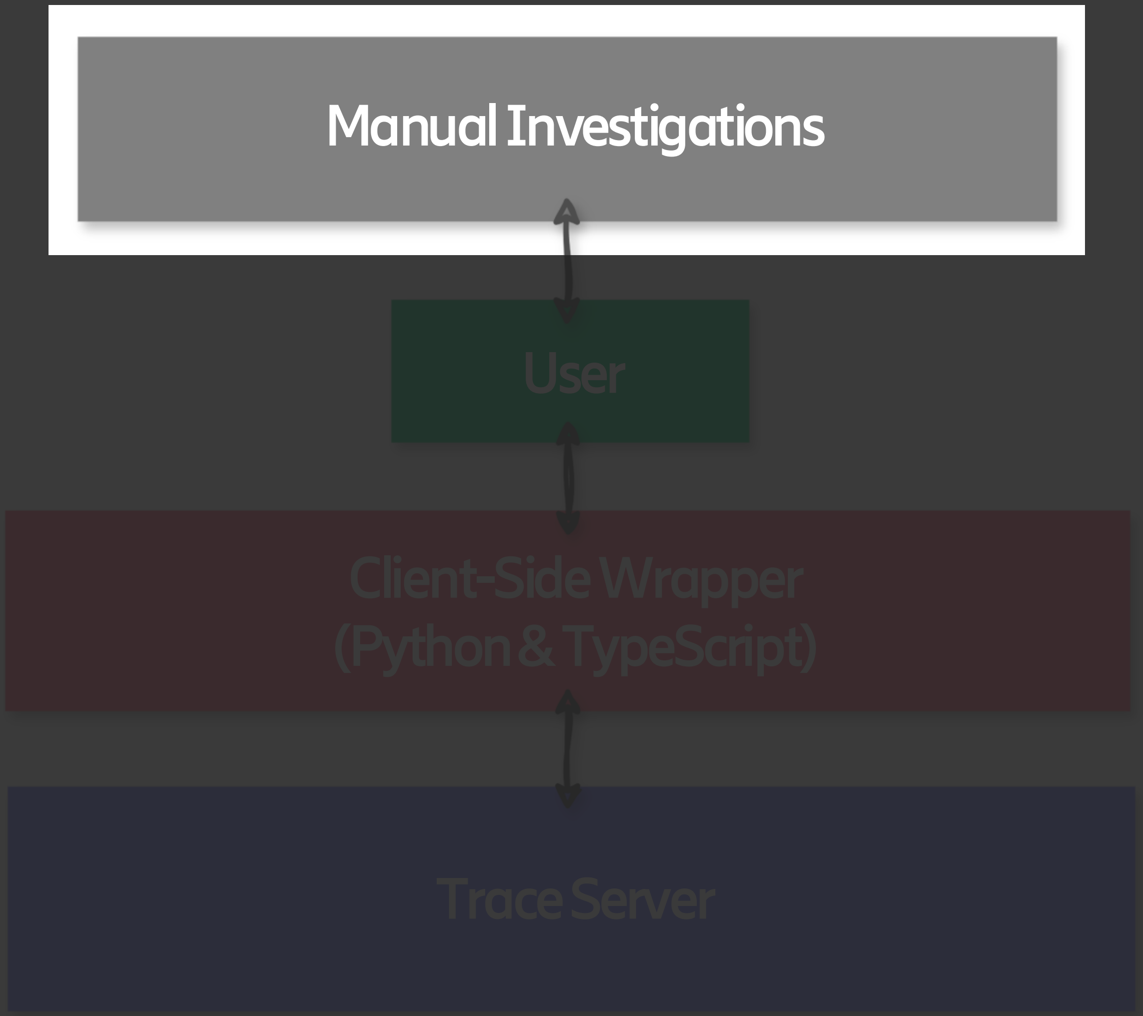
What areas can be investigated with TS?

- Identifying performance bottlenecks
  - CPU usage, disk I/O view, memory usage, scheduler and system call latency
- Diagnosing synchronization issues
  - Futex contention and IRQ latency
- Analyzing system and application behavior
  - Events table, flame chart call stack, thread and resources statuses
- Optimizing memory usage
  - Memory usage, memory latency
- Detailed latency analysis
  - Futex contentions, IRQ, schedulers, system calls, call stack

What is the procedure right now?



What is the procedure right now?



Issues and Downsides

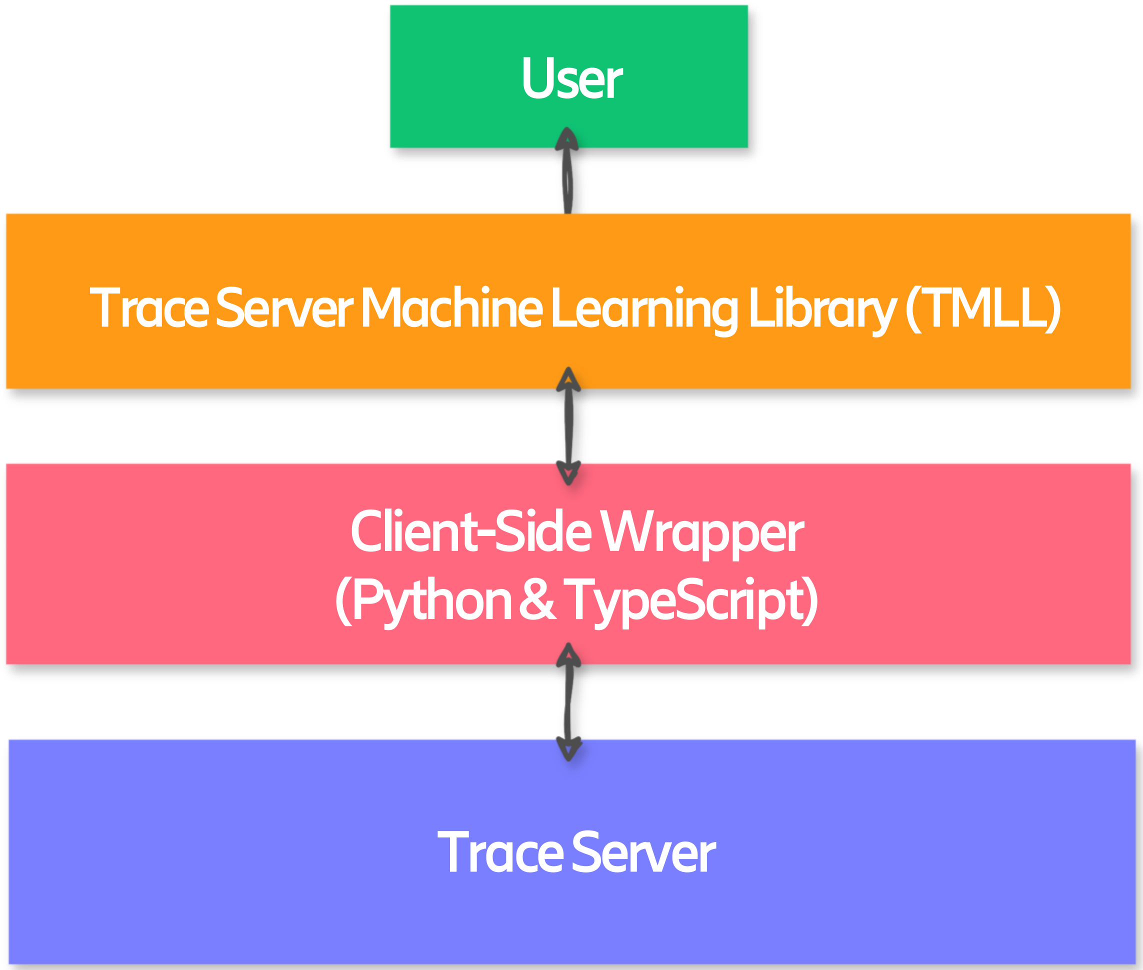
Trace Server

- User should know How to Get
- User should know What to Get

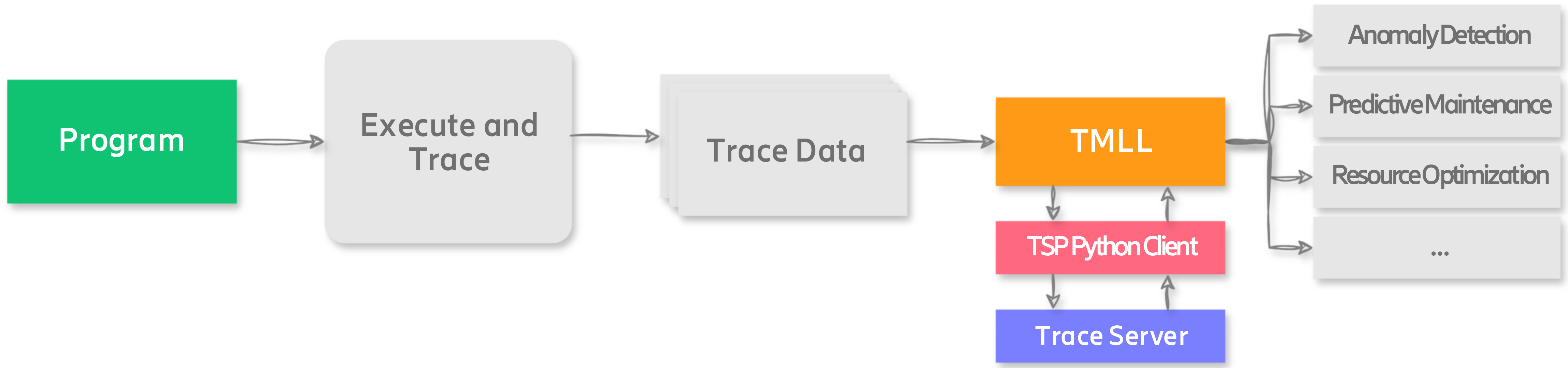
AI/ML

- User should know What to Use
- User should know How to Use
- User should know How to Analyze

What are we proposing?



TMLL Overview



TMLL Expected Features

Anomaly Detection

- Identify unusual patterns in system
- Provide potential regions of interest

Resource Optimization

- Optimize system resource allocation
- Analyze which areas can be optimized

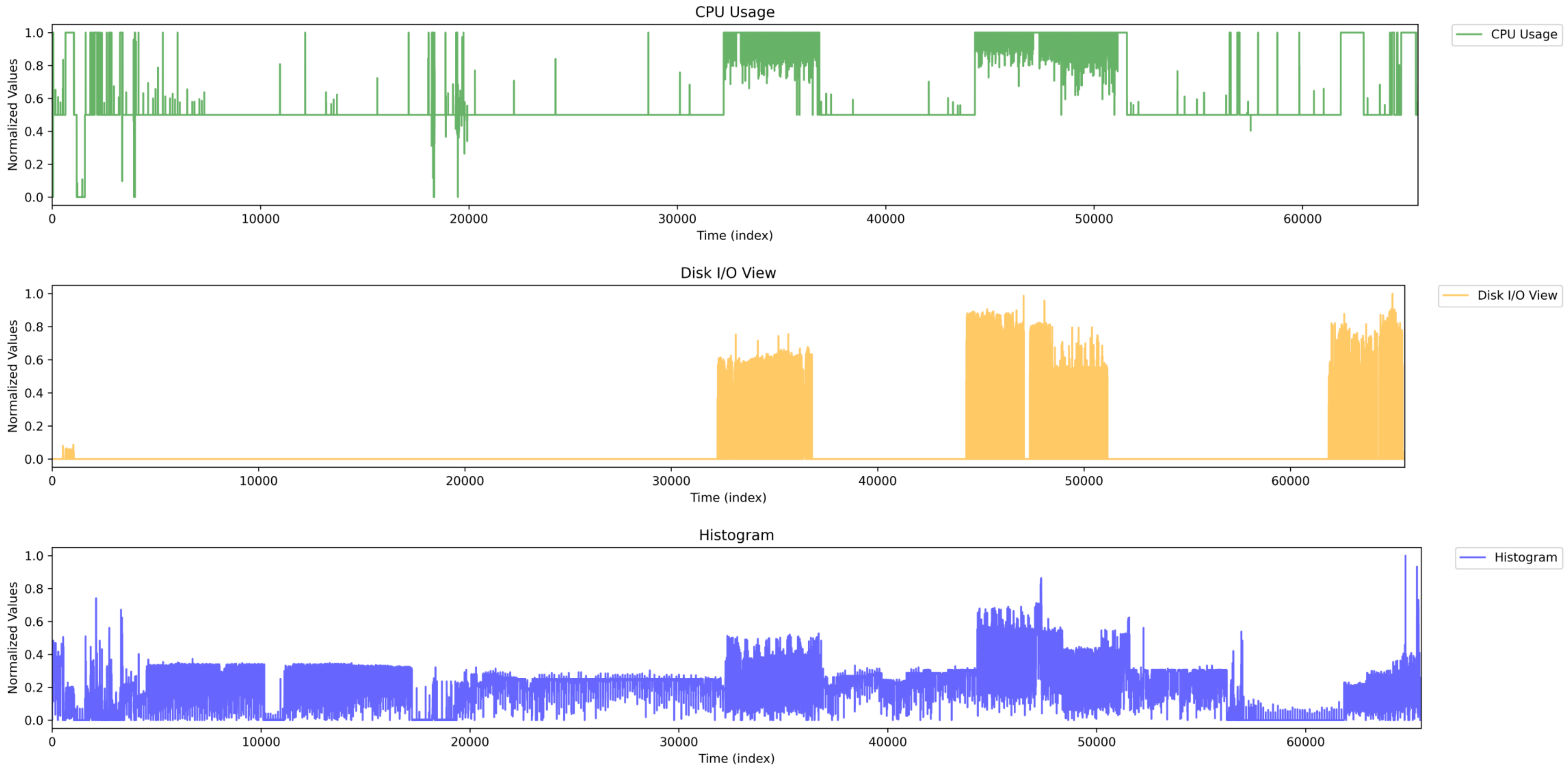
Predictive Maintenance

- Predict and prevent system failures
- Done in real-time analysis scenarios

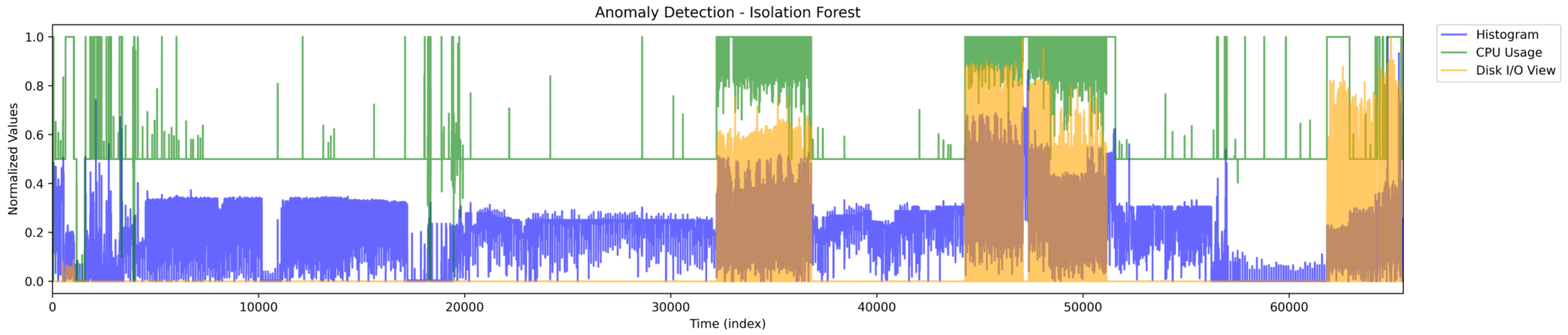
Performance Trend Analysis

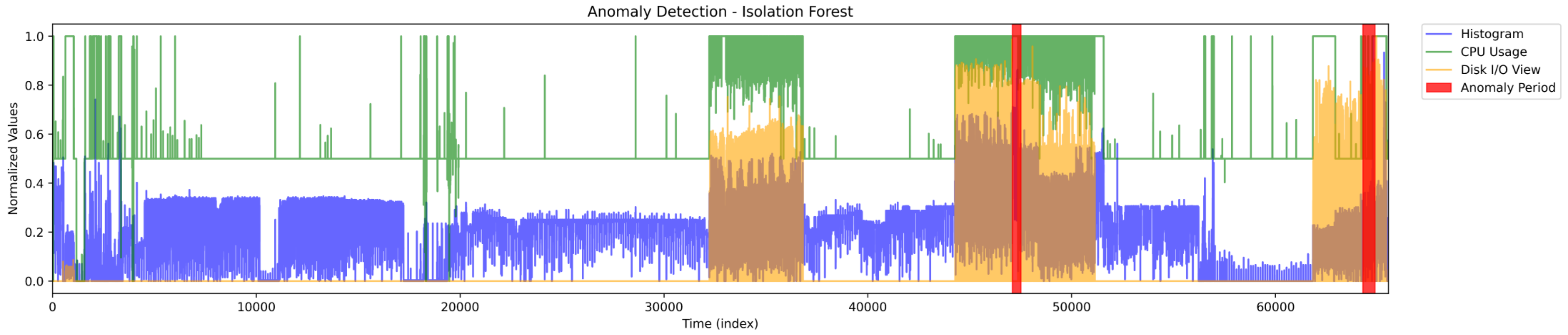
- Understand long-term performance trends
- Provide statistical insights on how system evolves

Anomaly Detection



Anomaly Detection (cont'd)

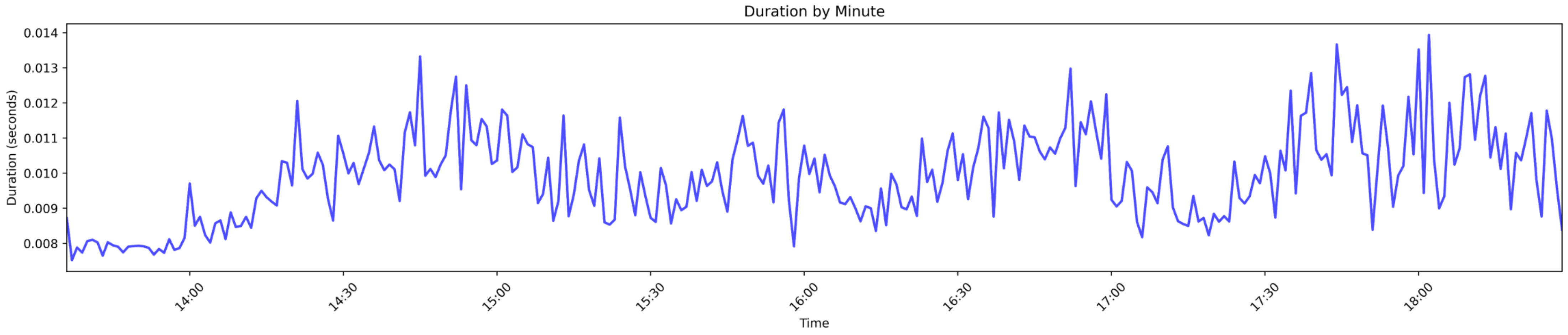


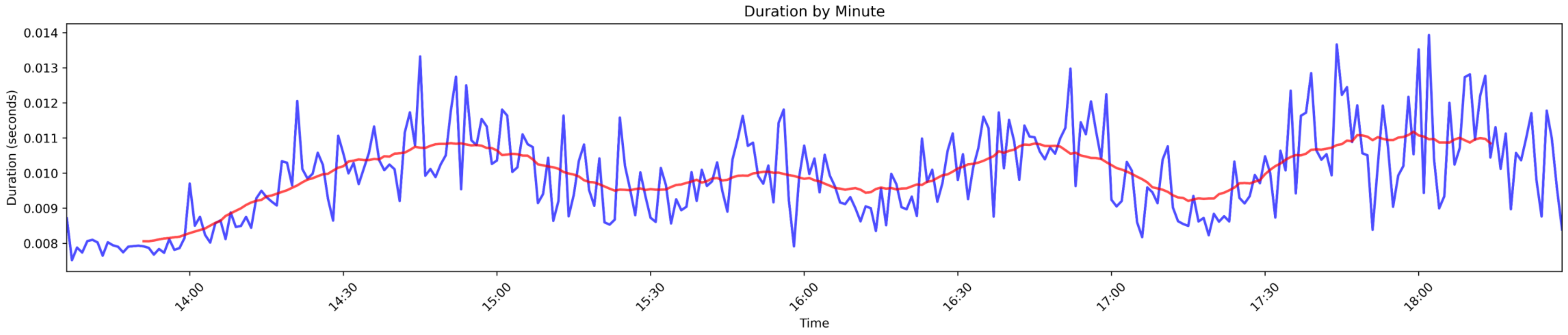


Potential Anomaly Period 1: 2024-04-24 02:13:**11.196**169984 to 2024-04-24 02:13:**11.210**216704

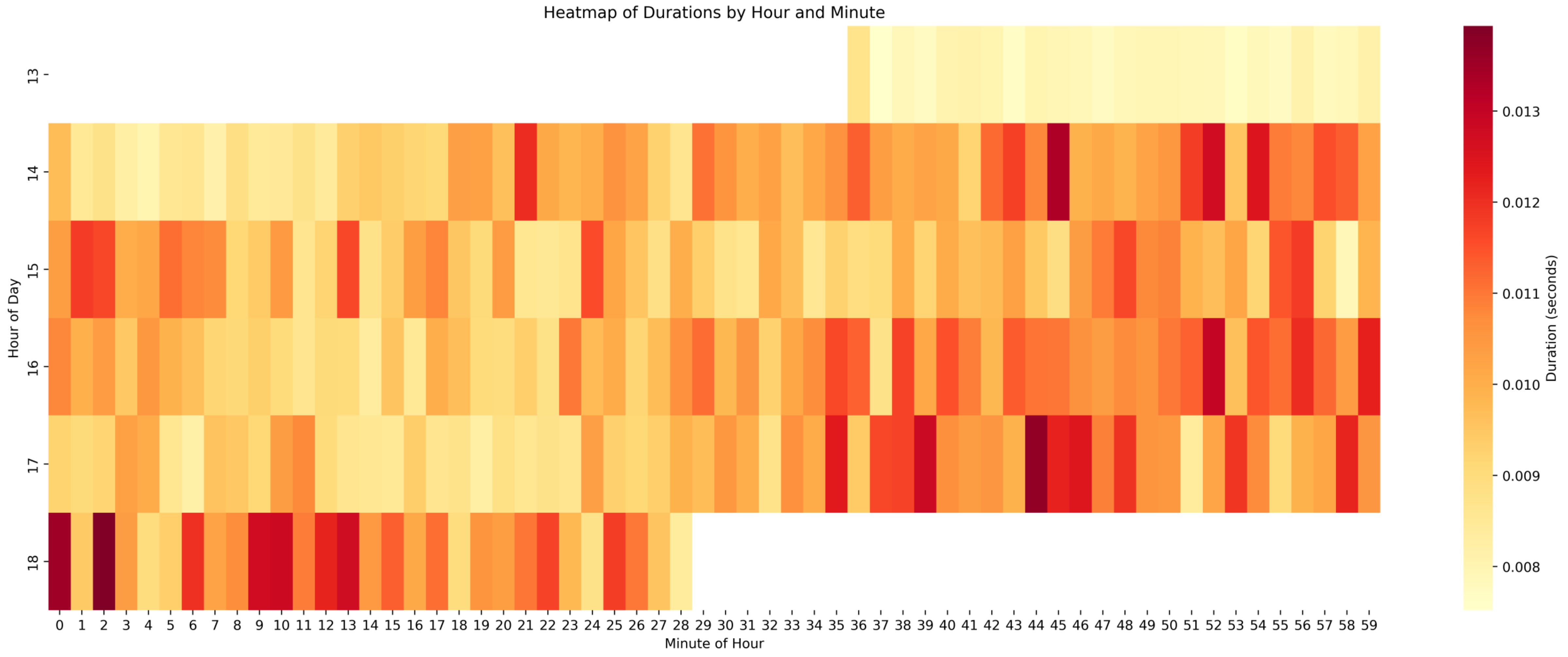
Potential Anomaly Period 2: 2024-04-24 02:13:**11.783**877888 to 2024-04-24 02:13:**11.804**144640

Seasonality Analysis

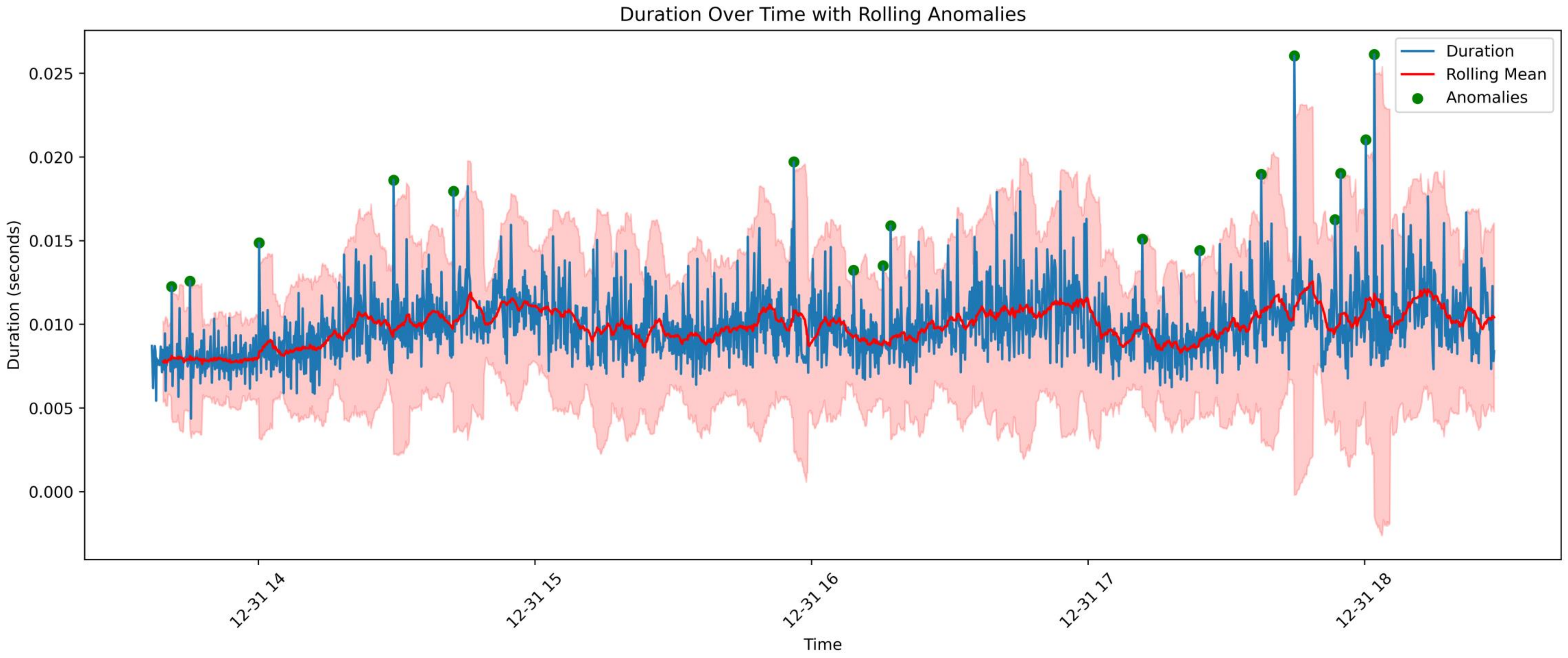




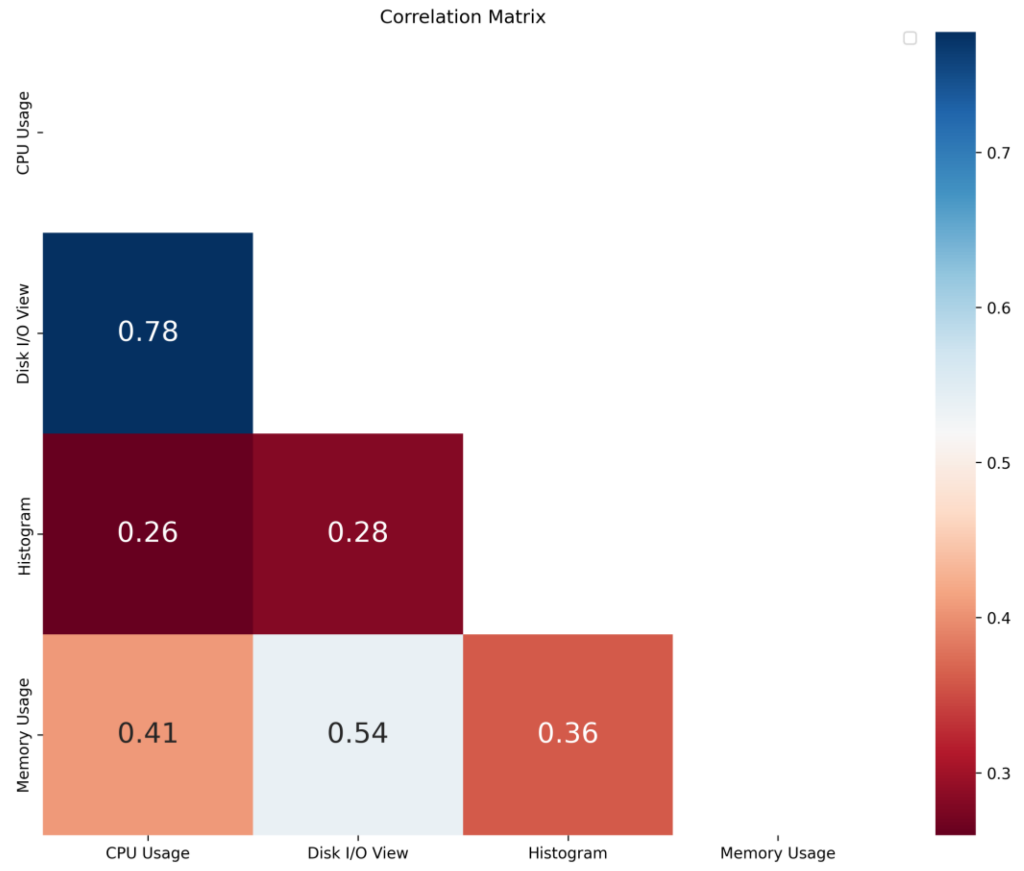
Seasonality Analysis (cont'd)

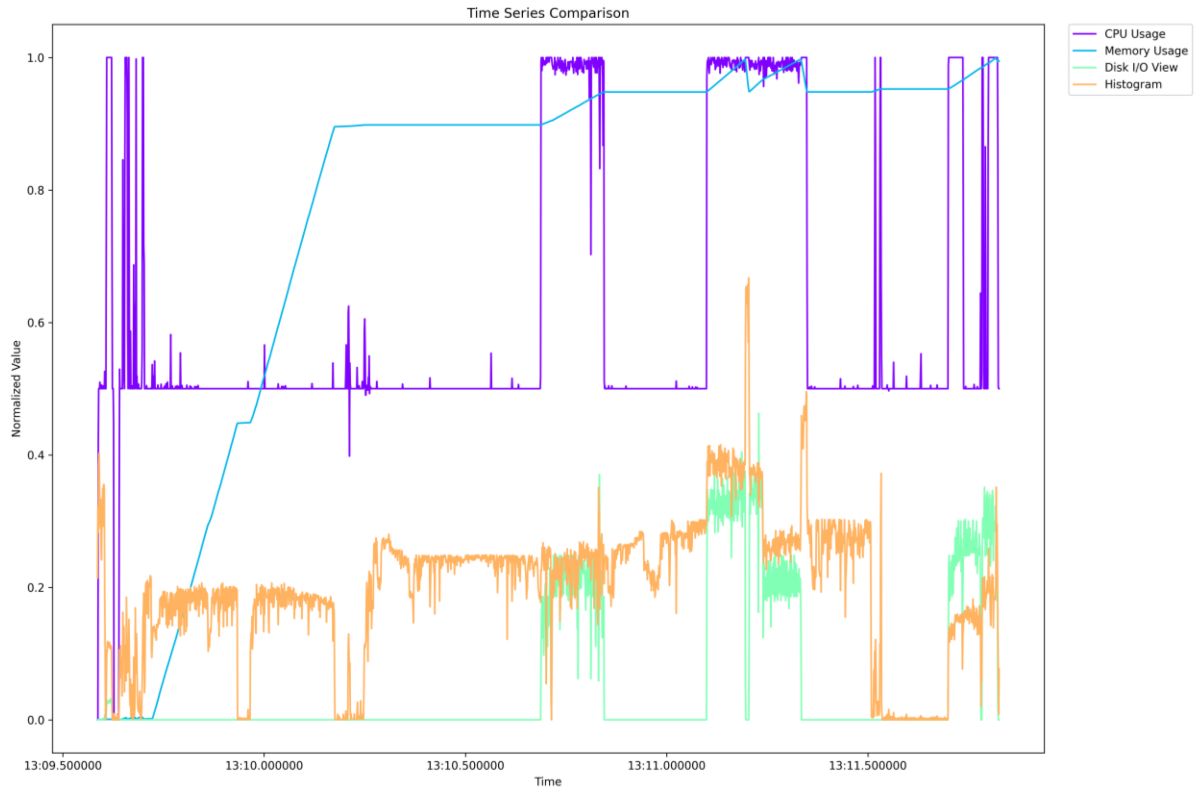


Seasonality Analysis (cont'd)

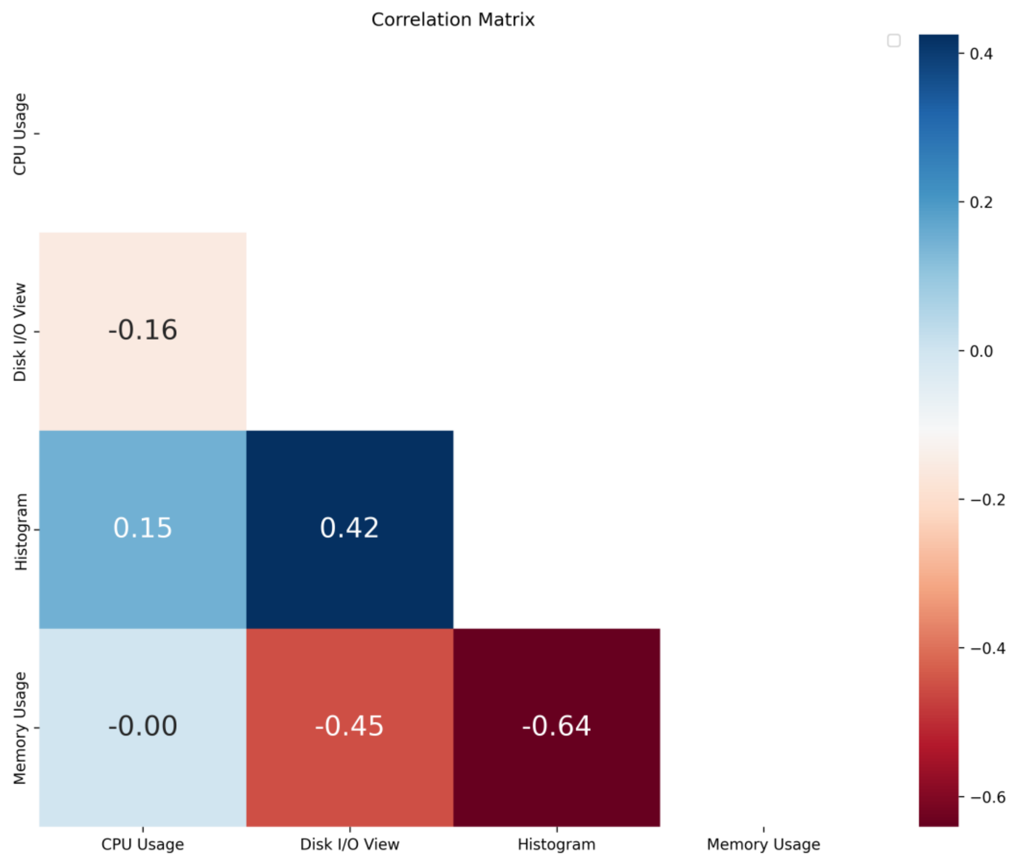


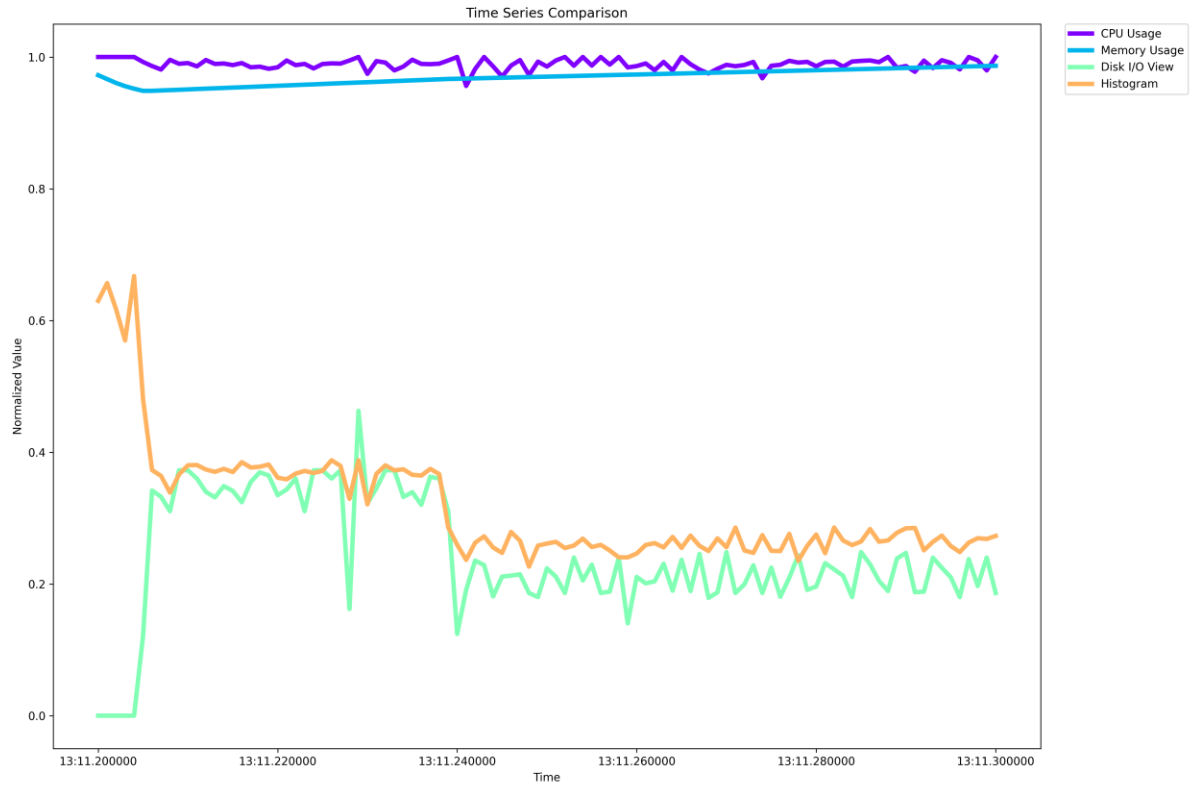
Correlation Analysis





Correlation Analysis (cont'd)

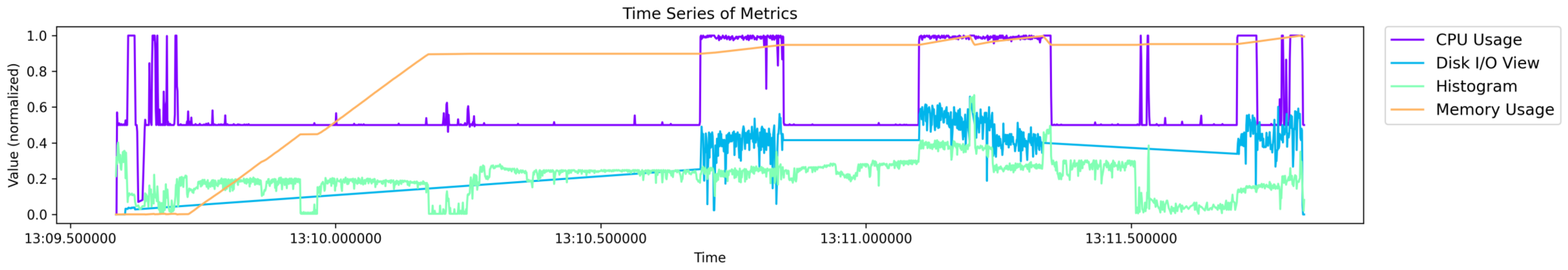




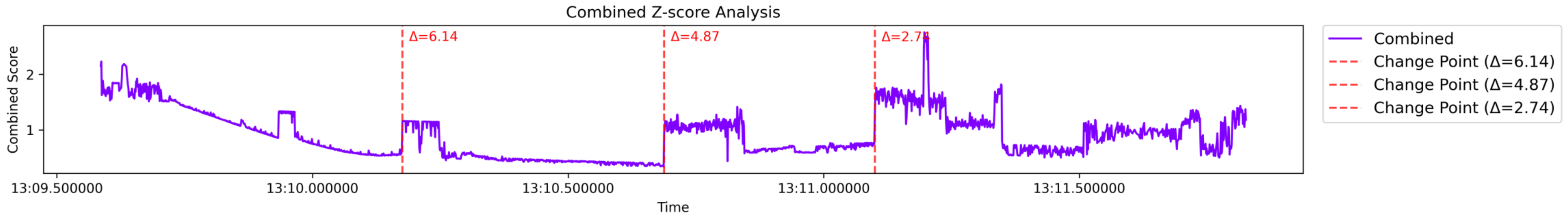
Start: 2024-04-24 02:13:**11.200**

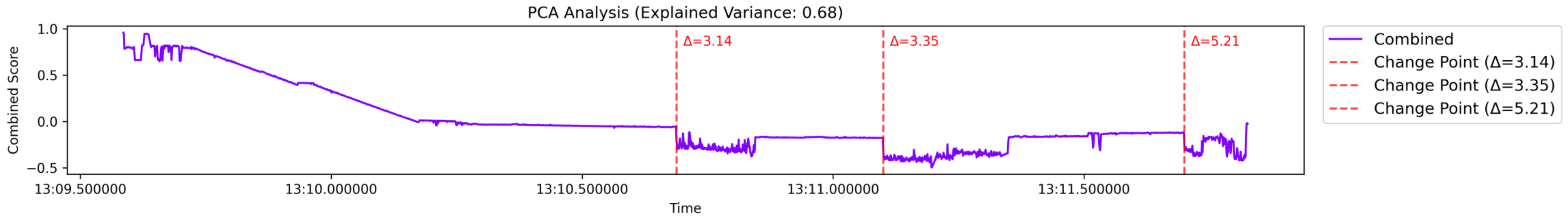
End: 2024-04-24 02:13:**11.300**

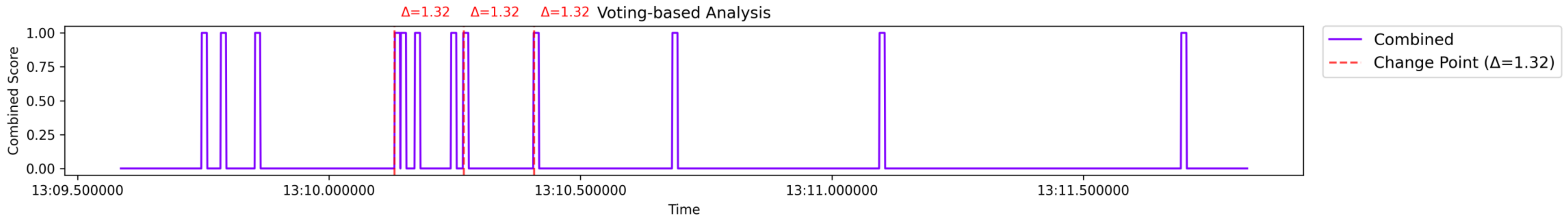
Change Point Detection



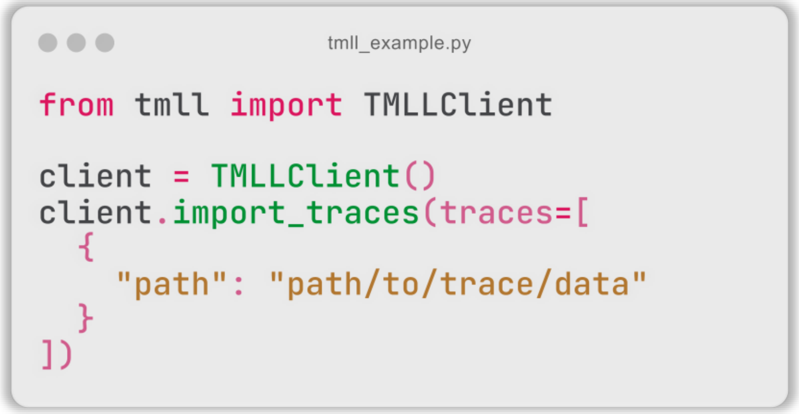
Change Point Detection (cont'd)





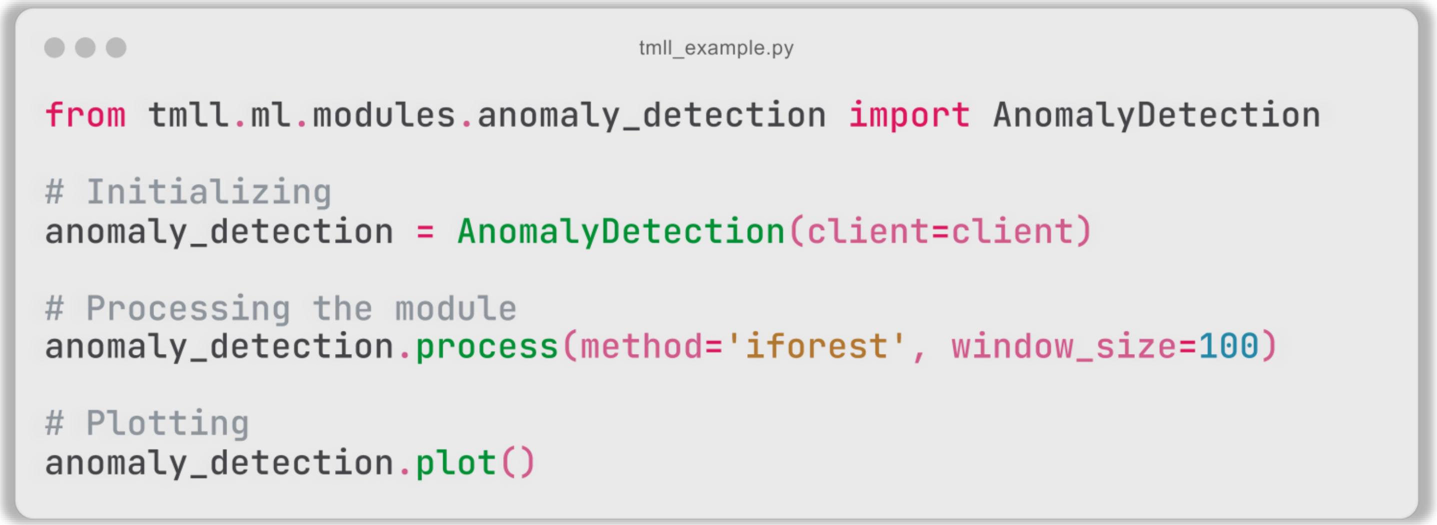


How is the Usage? (i.e., Coding)


tmll_example.py

```
from tmll import TMLLClient

client = TMLLClient()
client.import_traces(traces=[
  {
    "path": "path/to/trace/data"
  }
])
```


tmll_example.py

```
from tmll.ml.modules.anomaly_detection import AnomalyDetection

# Initializing
anomaly_detection = AnomalyDetection(client=client)

# Processing the module
anomaly_detection.process(method='iforest', window_size=100)

# Plotting
anomaly_detection.plot()
```