prmon Memory Anomaly Detection

Warm-up Project

Agnidipto Banik

February 28, 2026

Objective

- Generate memory time-series data using `prmon`
- Inject a controlled memory anomaly using burner tests
- Apply statistical anomaly detection
- Evaluate detection performance quantitatively and visually

Experimental Setup

- Environment: WSL Ubuntu + prmon (built from source)
- Metric used: RSS (Resident Set Size)
- Three segments:
  - Normal baseline
  - Elevated memory usage (anomaly)
  - Recovery to baseline
- Total samples: 228

Anomaly Injection

- Normal burner run establishes baseline behavior
- Memory parameters increased to create sustained elevation
- System returned to baseline to simulate recovery
- Anomaly region labeled using known injection interval

Data Generation Using prmon

```
# Baseline run
~/prmon/build/package/prmon \
  --interval 1 \
  --filename data/normal_part.csv \
  -- ~/prmon/build/package/tests/mem-burner \
  --malloc 500 --sleep 150

# Elevated memory (anomaly)
~/prmon/build/package/prmon \
  --interval 1 \
  --filename data/anomaly_part.csv \
  -- ~/prmon/build/package/tests/mem-burner \
  --malloc 650 --sleep 150

# Recovery run
~/prmon/build/package/prmon \
  --interval 1 \
  --filename data/segment3_recovery.csv \
  -- ~/prmon/build/package/tests/mem-burner \
```

Dataset Construction

```
s1 = pd.read_csv("../data/normal_part.csv", sep=r"\s+"
    )
s2 = pd.read_csv("../data/anomaly_part.csv", sep=r"\s+
    ")
s3 = pd.read_csv("../data/segment3_recovery.csv", sep=
    r"\s+")

df = pd.concat([s1, s2, s3], ignore_index=True)

df["is_anomaly"] = 0
df.loc[len(s1):len(s1)+len(s2)-1, "is_anomaly"] = 1
```

- Concatenates normal, anomaly, and recovery segments
- Ground truth labels created using known injection interval
- Enables quantitative evaluation

Rolling Z-Score Implementation

```
window = 30
threshold = 3

df["rolling_mean"] = df["rss"].rolling(window=window).
    mean()
df["rolling_std"] = df["rss"].rolling(window=window).
    std()

df["z_rolling"] = (
     (df["rss"] - df["rolling_mean"]) /
     df["rolling_std"].replace(0, np.nan)
).fillna(0)

df["rolling_detected"] = (df["z_rolling"].abs() >
    threshold).astype(int)
```

- Computes rolling mean and standard deviation
- Standardizes deviation using Z-score
- Flags $|z| > 3$ as anomaly

Frozen Baseline Implementation

```
baseline_mean = s1["rss"].mean()
baseline_std = s1["rss"].std()

df["z_frozen"] = (df["rss"] - baseline_mean) /
    baseline_std
df["frozen_detected"] = (df["z_frozen"].abs() >
    threshold).astype(int)
```

- Baseline statistics computed from clean segment
- Fixed reference used for all future samples
- Detects sustained structural shifts
- Does not adapt to new regimes

Evaluation Metrics

```
precision_score(df["is_anomaly"], df[col])
recall_score(df["is_anomaly"], df[col])
f1_score(df["is_anomaly"], df[col])
confusion_matrix(df["is_anomaly"], df[col])
```

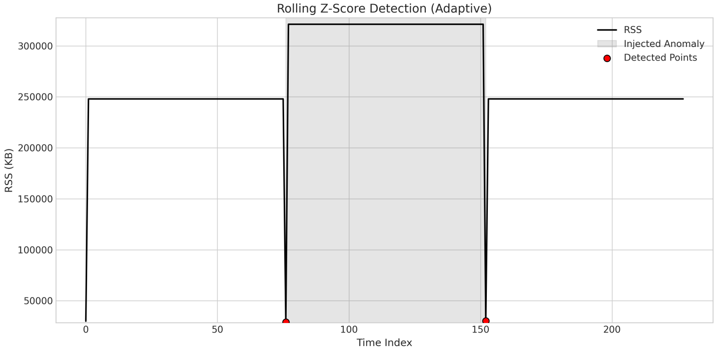
- Precision: false positive control
- Recall: anomaly coverage
- F1-score: balance between precision and recall
- Confusion matrix provides detailed breakdown

Rolling Z-Score (Adaptive Detection)

$$z_t = \frac{x_t - \mu_{window}}{\sigma_{window}}$$

- Compares current value to recent behavior
- Adapts dynamically to new levels
- Threshold: $|z| > 3$

Rolling Z-Score – Results



Why Rolling Fails

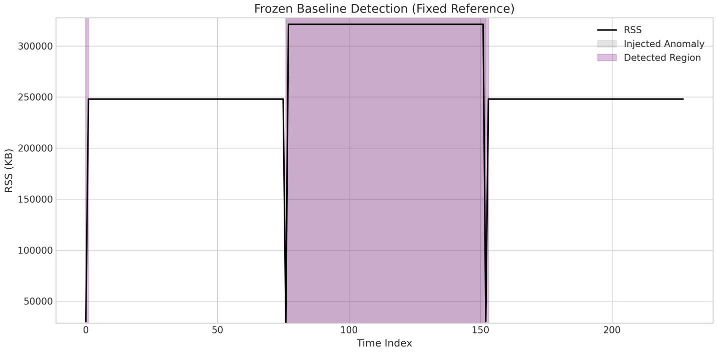
- Detects boundary transitions only
- Rolling window adapts to sustained shift
- Elevated regime becomes statistically normal
- Low recall for step anomalies

Frozen Baseline Z-Score (Fixed Reference)

$$z_t = \frac{x_t - \mu_{baseline}}{\sigma_{baseline}}$$

- Baseline statistics computed from initial normal segment
- No adaptation to new regimes
- Detects sustained structural deviation

Frozen Baseline – Results



Why Frozen Performs Better

- Step anomaly introduces sustained mean shift
- Frozen model remains anchored to healthy baseline
- Entire elevated regime remains statistically significant
- High precision and recall observed

Final Quantitative Results

Rolling Z-Score

$$\begin{bmatrix} 151 & 1 \\ 75 & 1 \end{bmatrix}$$

Frozen Baseline

$$\begin{bmatrix} 150 & 2 \\ 0 & 76 \end{bmatrix}$$

Why Not Machine Learning?

- Warm-up focused on controlled statistical evaluation
- Dataset is small and structured
- Statistical methods are interpretable and sufficient

Trade-offs and Sustainability

- Rolling suitable for short-lived spikes
- Frozen requires clean baseline period
- Frozen may struggle with gradual drift
- Real deployments require adaptive baselines and multi-metric monitoring

Conclusion

- Controlled memory anomaly successfully injected
- Statistical detection methods applied and evaluated
- Frozen baseline outperformed rolling detection
- Demonstrates structured experimental reasoning