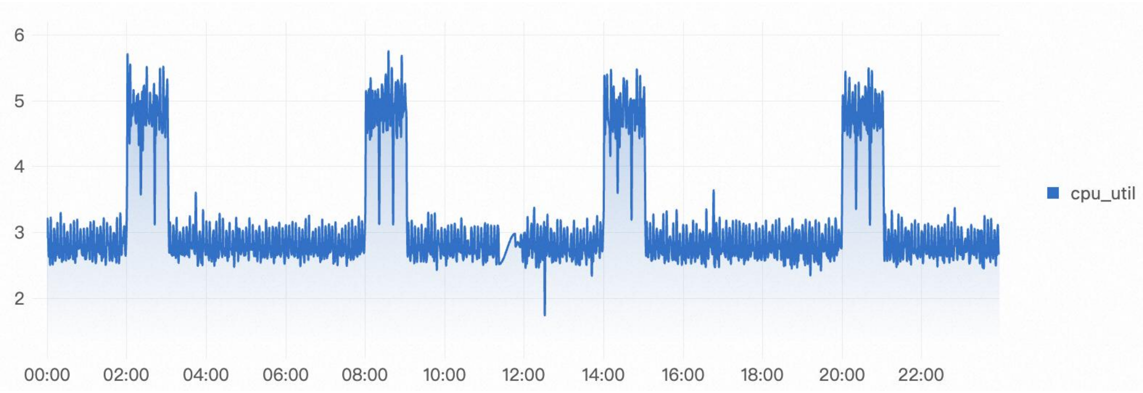
BacktrackSTL: Ultra-Fast Online Seasonal-Trend Decomposition with Backtrack Technique

Haoyu Wang, Hongke Guo, Zhaoliang Zhu, You Zhang, Yu Zhou, Xudong Zheng
Alibaba Cloud, Alibaba Group

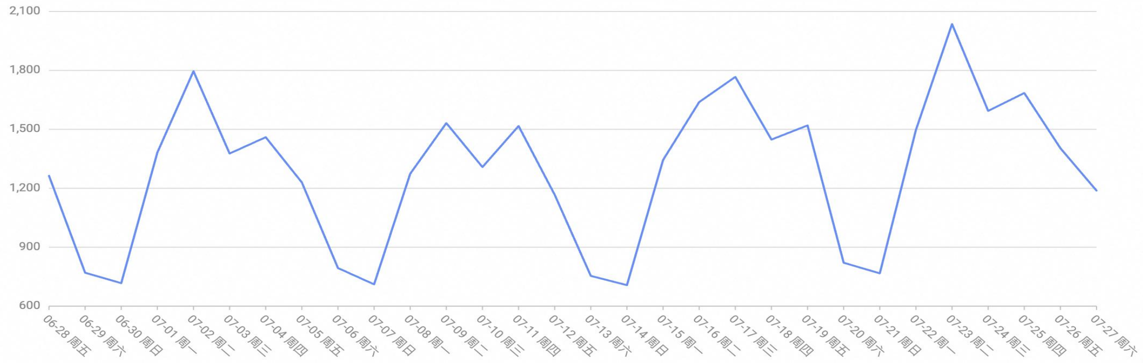
ACM SIGKDD 2024

Background

- **Periodicity:** time series which exhibit repeating segments.
  - CPU usage (6-hour period):
    - Scheduled Task



  - Page views of web (weekly):
    - Human activity



  - …

Problem Statement

- Online seasonal-trend decomposition (STD)
  - STD decomposes a periodic time series into trend, seasonality, and residual components.
  - In online scenario, decomposition is incremental with limited memory.

$$y_t = \tau_t + s_t + r_t$$

Efficiency Challenge

- In Alibaba Cloud, **time efficiency** has the highest priority in the design of STD.
  - Save computational cost
  - Reduce response latency
  - …

1,000,000,000+ Metrics

10,000,000+ VMs

5,000+ Clusters

30+ Regions

Related Work Comparison

- The complexities of existing algorithms still need improvement.

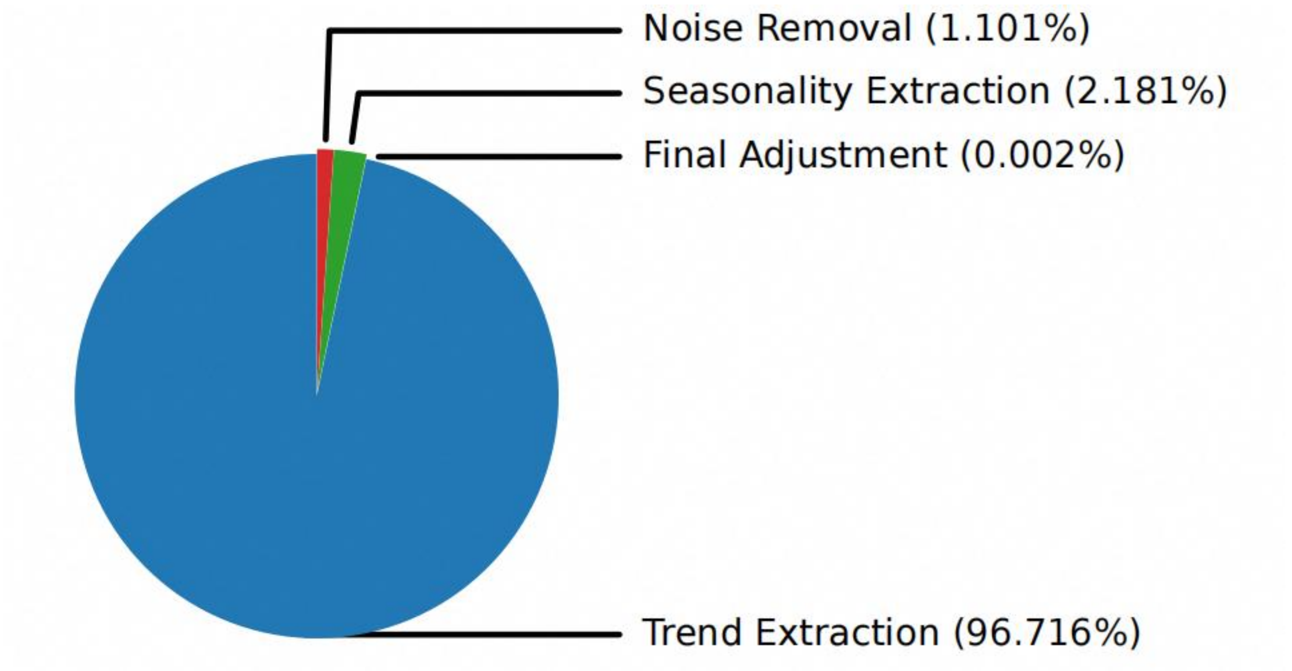
Algorithm	Trend Jump	Seasonality Shift	Outlier Tolerance	Online Complexity
STL	No	No	No	-
TBATS	Yes	No	No	-
STR	No	Yes	Yes	-
SSA	No	No	No	-
RobustSTL	Yes	Yes	Yes	-
OnlineSTL	No	No	No	$O(T)$
OneShotSTL	Yes	Yes	No	$O(I)$
BacktrackSTL	**Yes**	**Yes**	**Yes**	$\boldsymbol{O(1)}$

Contribution

- **Time efficiency**
  - BacktrackSTL is the first non-iterative online seasonal-trend decomposition algorithm with period-independent **O(1)** time complexity.
- **Accuracy**
  - BacktrackSTL combines outlier-resilient smoothing, non-local seasonal filtering and backtrack technique to achieve the **robustness to outlier, seasonality shift and trend jump**.
- **Deployment**
  - BacktrackSTL is deployed based on **Apache Flink** in the production environment of Alibaba Cloud for over a year.

Motivation

- Time cost analysis on RobustSTL
  - **Trend extraction** stage consumes the majority of the time, which solves an optimization problem based on the L1-norm.

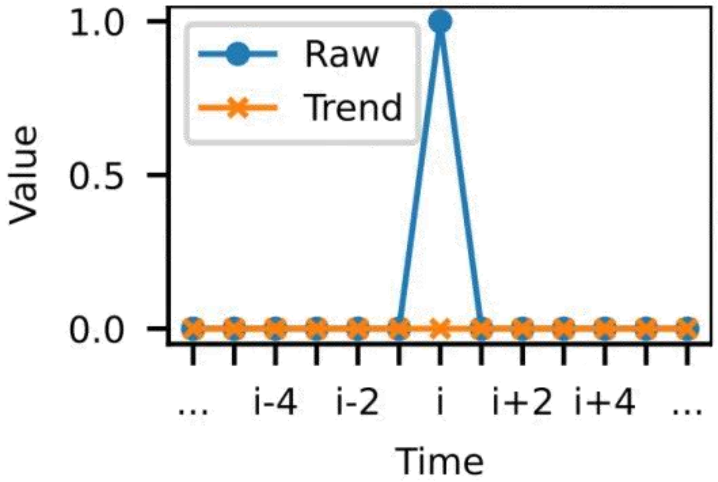


Motivation

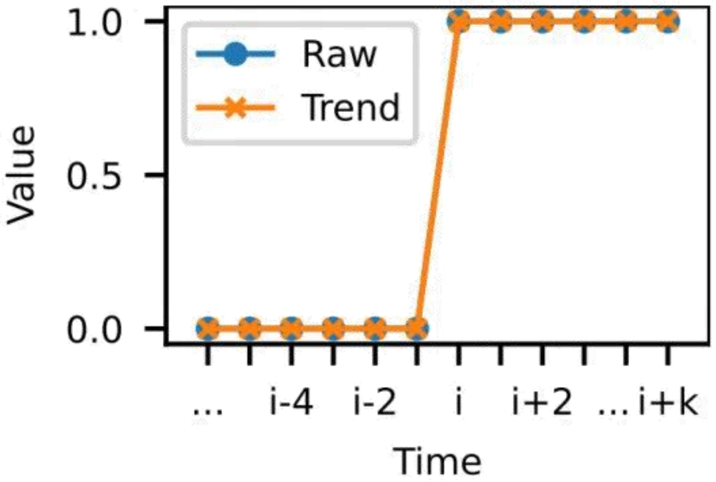
- Effectiveness of L1-norm optimization
  - Optimization goal

$$\min_{\tau_{1\ldots N}} \sum_{t=T+1}^{N} |(y_t - \tau_t) - (y_{t-T} - \tau_{t-T})| + \lambda_1 \sum_{t=2}^{N} |\tau_t - \tau_{t-1}| + \lambda_2 \sum_{t=3}^{N} |\tau_t - 2\tau_{t-1} + \tau_{t-2}|$$

  - Robust to both **outlier** and **trend jump**



(a) Outlier



(b) Trend jump

Motivation

High Complexity

L1-norm optimization against both outlier and trend jump

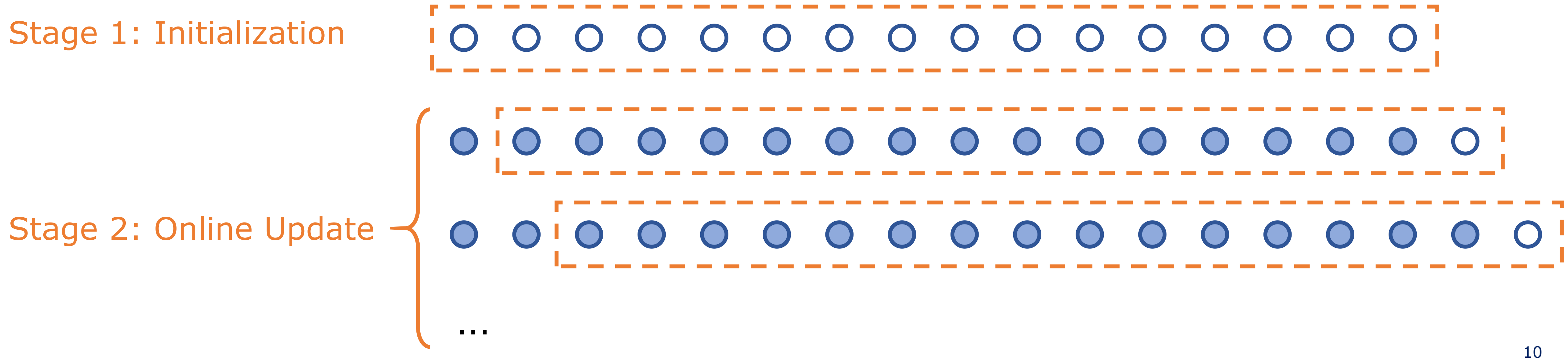
Low Complexity

Combination

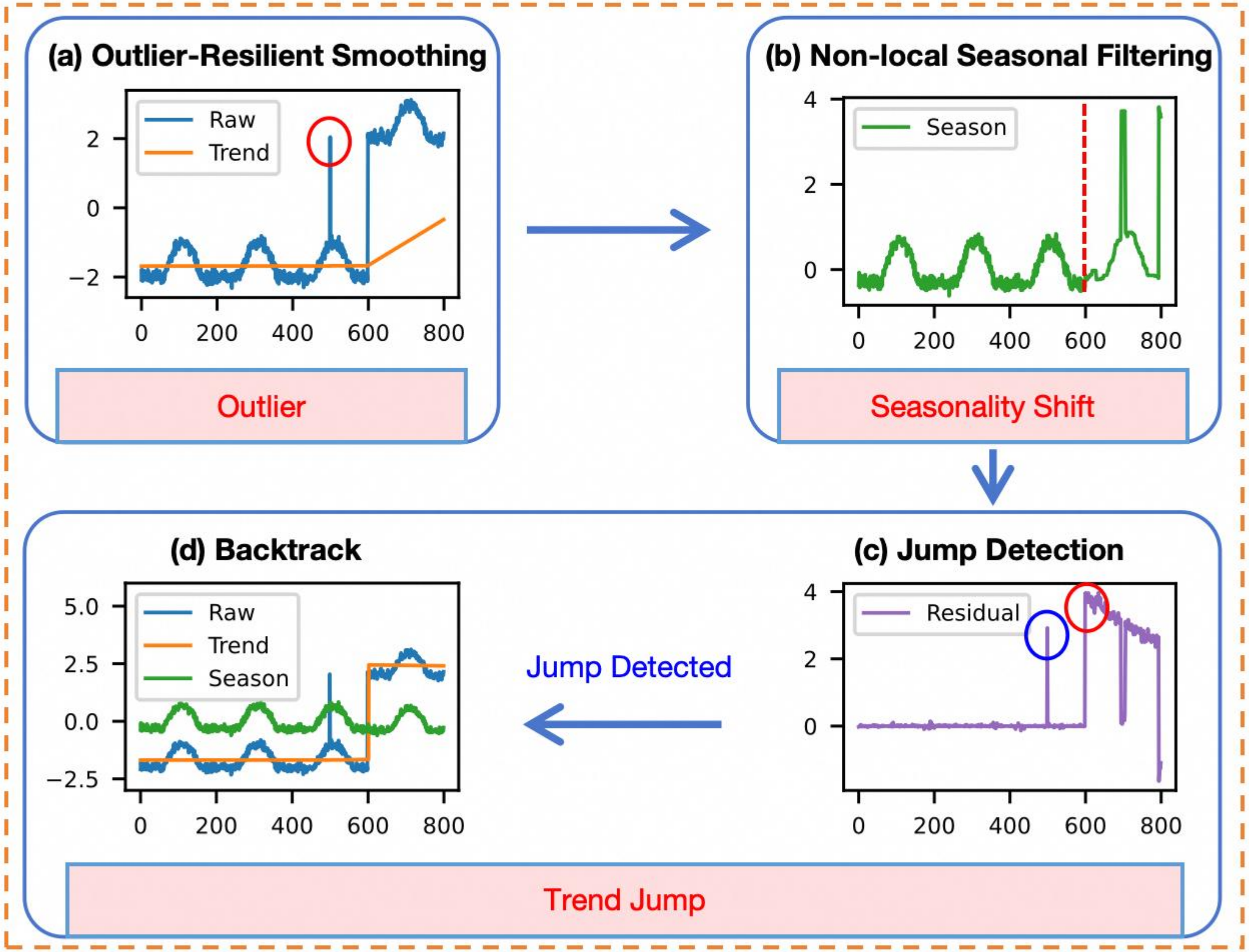
- Simple method against outlier
- Simple method against trend jump

BacktrackSTL

- Limited memory => Sliding window
  - Window size: W=(K+1)T
- Decomposition history => Initialization + Online update
  - Initialization stage decomposes all values in the first window (conducts **once**)
  - Online update stage decomposes the last value in the window with the history



Online Update



Outlier-Resilient Smoothing

- Moving average smoothing

$$\tau_t = \frac{1}{W}\sum_{i=t-W+1}^{t} y_i = \frac{1}{W}\sum_{i=t-W+1}^{t}(\tau_i + s_i + r_i)$$

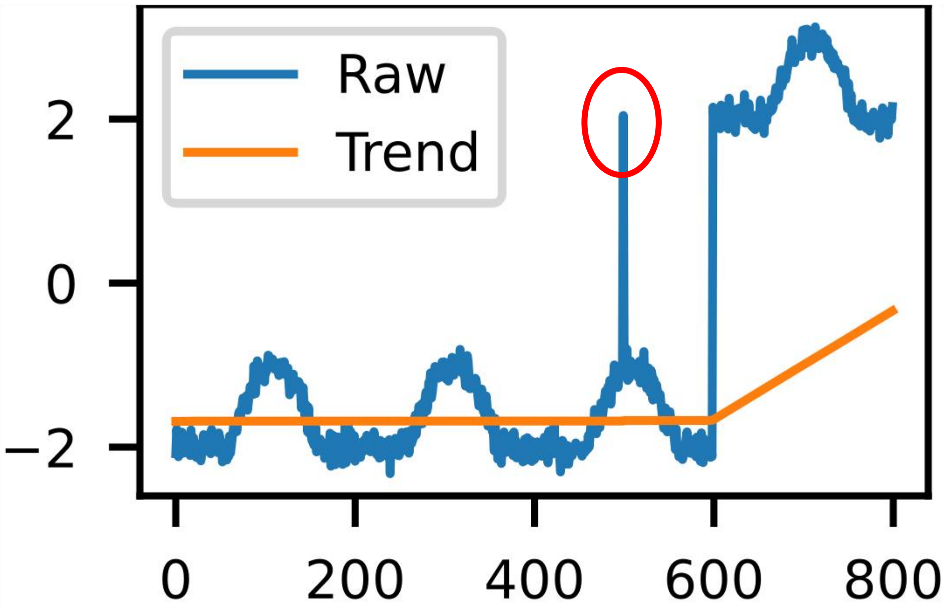
- Outlier-resilient mechanism
  - Detect with dynamic N-Sigma
  - Repair to reference value

$$\hat{y}_t = \tau_{t-1} + \arg\min_{s_i, i\in\Omega} |s_i - (y_t - \tau_{t-1})|$$

$$\Omega = \{i | (t' = t - kT, i = t' \pm h)\}$$

$$k = 1, 2, \ldots, K; h = 0, 1, \ldots, H$$

- Time complexity: O(KH) ~ O(1)



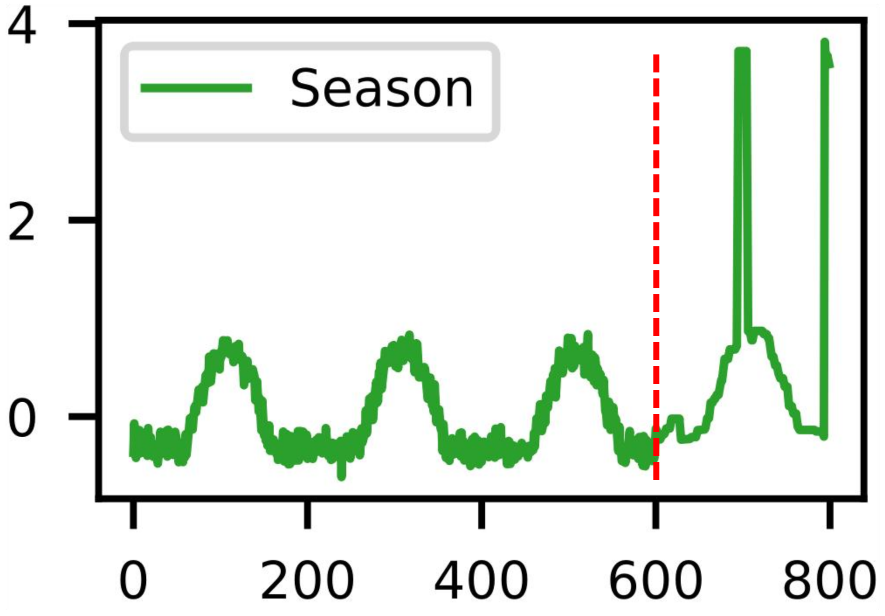
Non-local Seasonal Filtering

- Proposed by RobustSTL (AAAI 19)
- Robustness to seasonality shift

$$s_t = \sum_{j \in \Omega} w_j^t y_j'$$

$$w_j^t = \frac{1}{z} \exp\{-\frac{(j-t')^2}{2H^2} - \frac{(y_j' - y_t')^2}{2\delta^2}\}$$

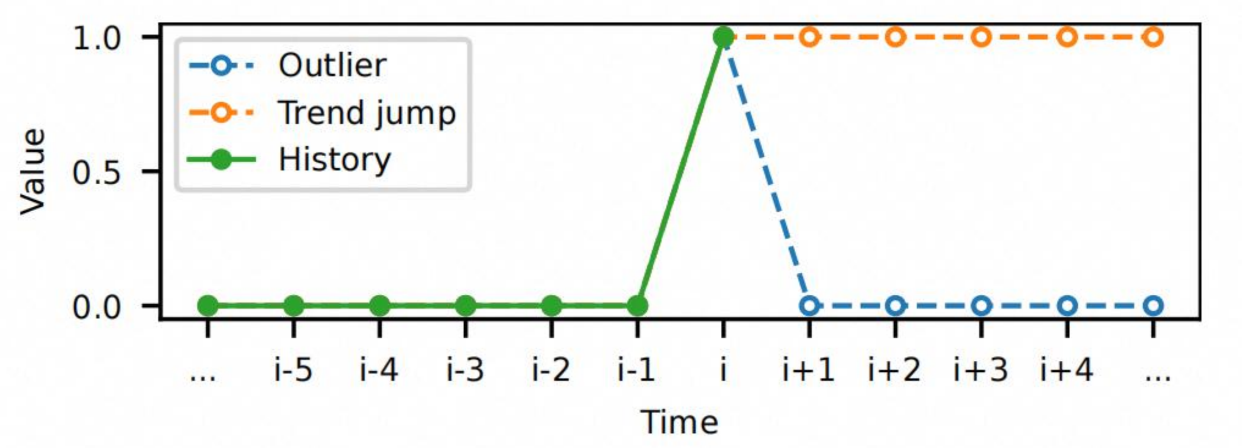
$$y_j' = y_j - \tau_j$$



- Time complexity: O(KH) ~ O(1)

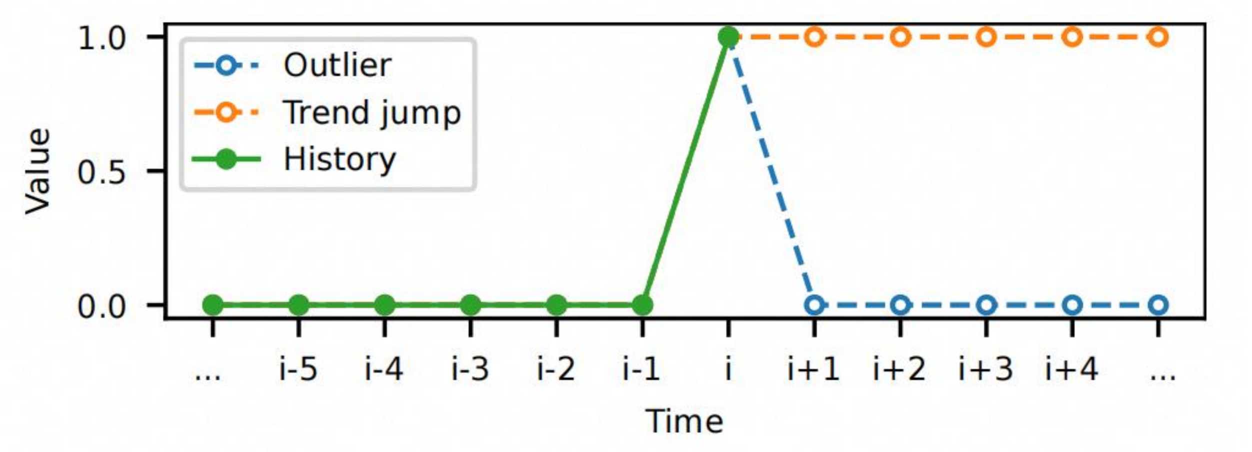
Jump Detection

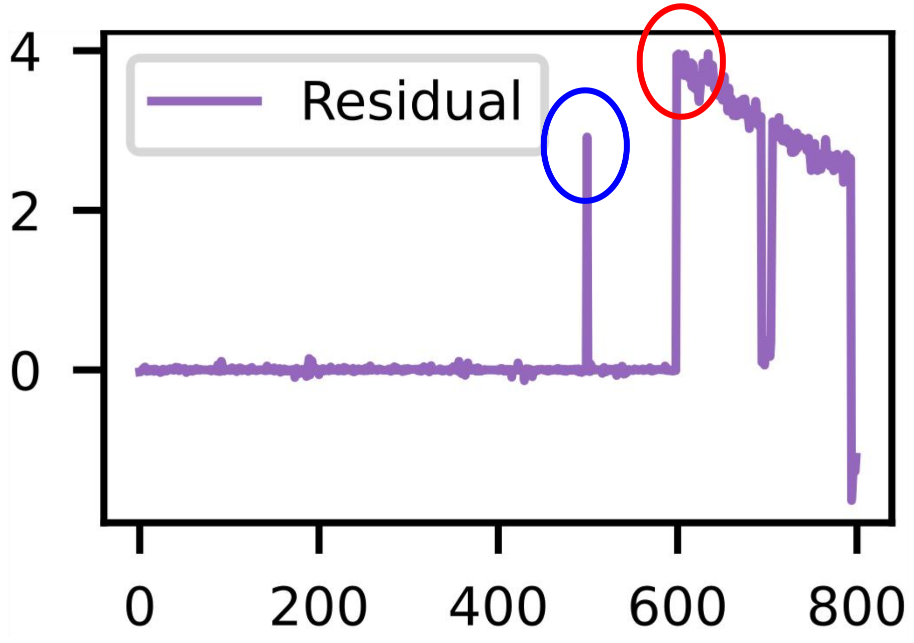
- Decision is naturally **delayed** in online scenario



Jump Detection

- Decision is naturally **delayed** in online scenario





- Jump leads to **consecutive high residuals**
- Time complexity: O(1)

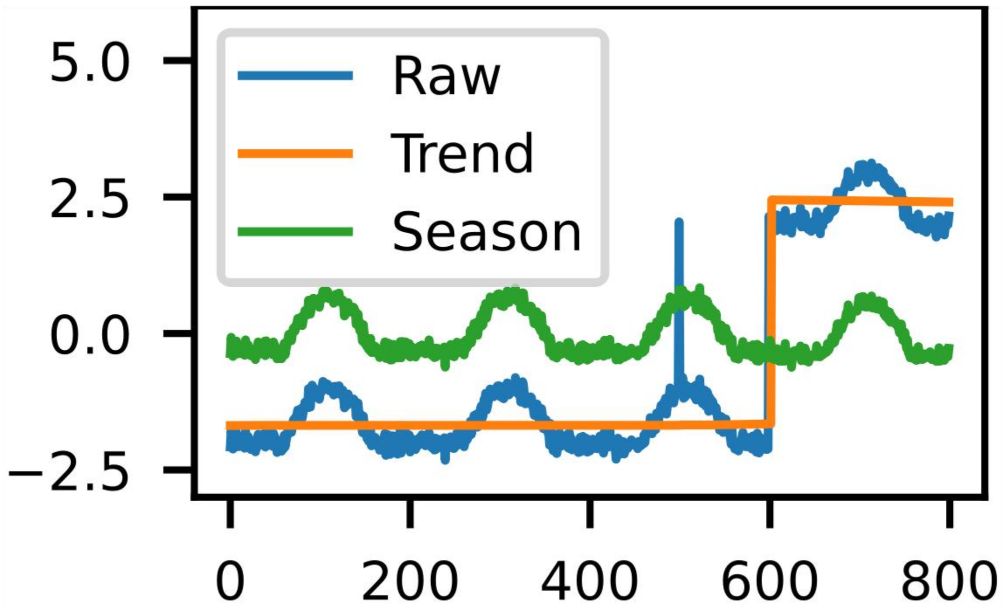
Backtrack

- Correct decomposition after a jump
  - All trends are estimated as a constant

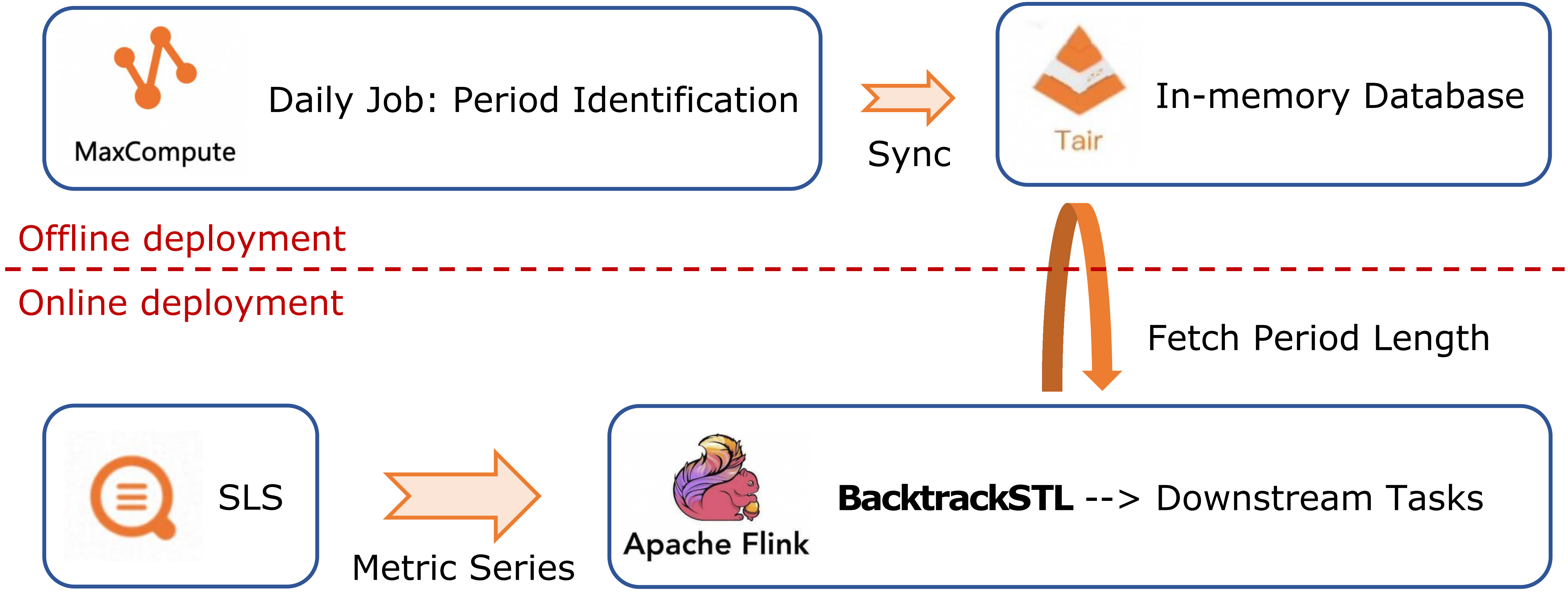
$$\bar{\tau} = \frac{1}{L} \sum_{i=t-L+1}^{t} y_i - s_{i-T}$$

  - Seasonal components are estimated with non-local seasonal filtering

- Only conduct when a jump is detected
  - Extremely low frequency
  - Almost no influence on time complexity



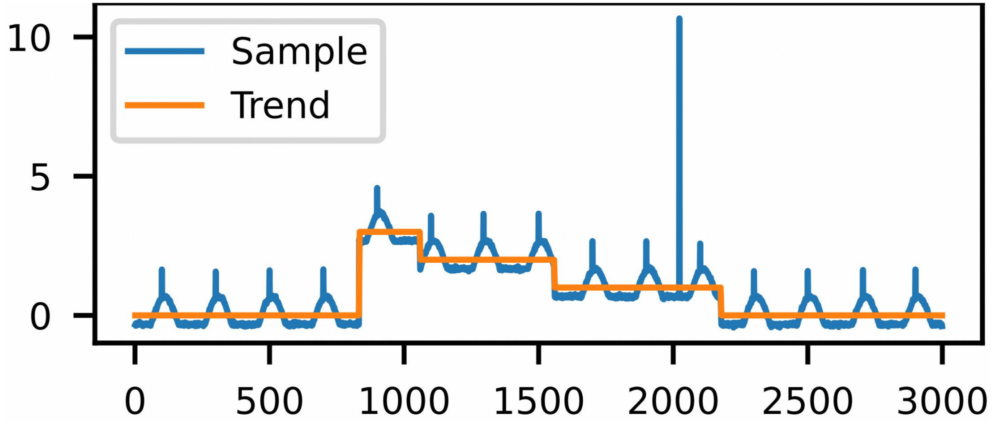
Deployment



MaxCompute
Daily Job: Period Identification
Sync
Tair
In-memory Database
Offline deployment
Online deployment
Fetch Period Length
SLS
Metric Series
Apache Flink
BacktrackSTL --> Downstream Tasks

Experiment Setup

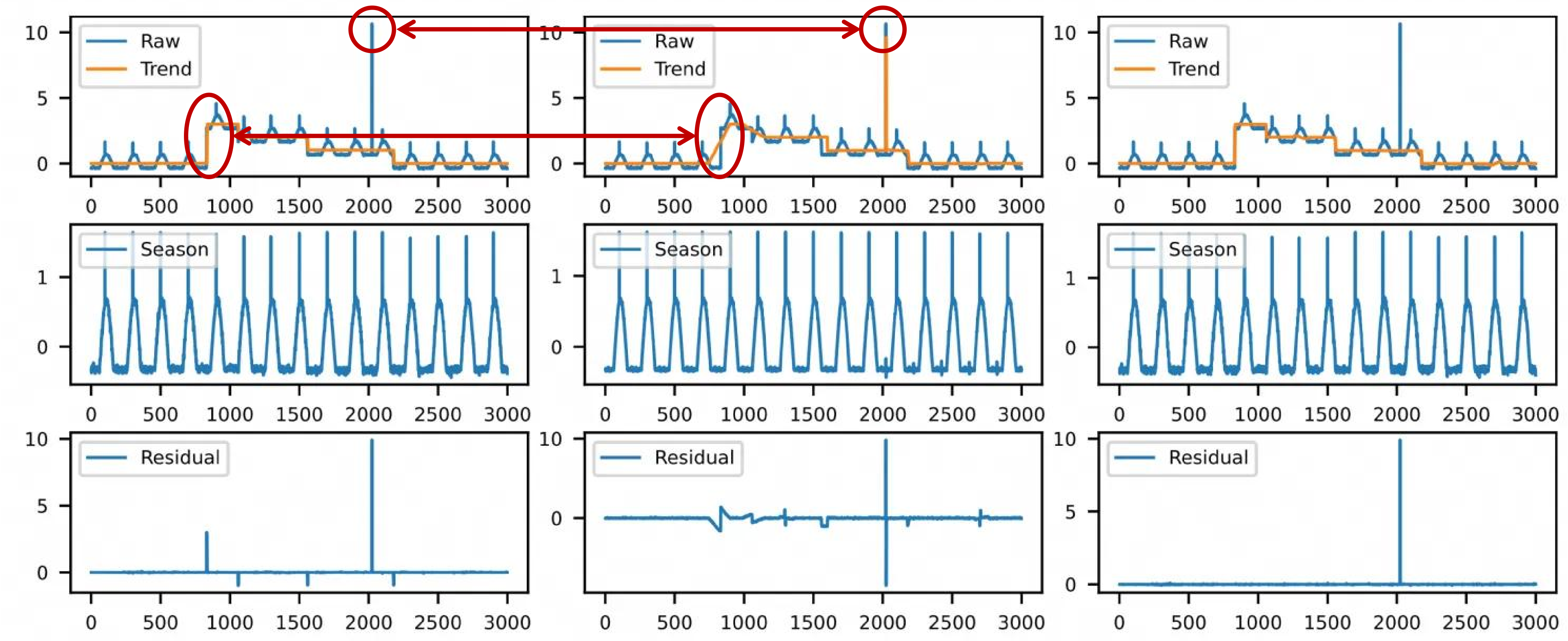
- Environment
  - VM: ecs.re4.10xlarge (40 vCPU cores + 480 GiB memories)
  - OS: 64-bit CentOS 7.9
- Synthetic Dataset
  - Series length 3000 (T=200)
  - 4 trend jumps and 1 severe outlier
  - Seasonal shifts with a maximum of 5



- Algorithm
  - Offline: STL, SSA, TBATS, **RobustSTL**
  - Online: Window-STL, Window-SSA, Window-TBATS, Window-RobustSTL, Online-RobustSTL, OnlineSTL, **OneShotSTL**, **BacktrackSTL**

Accuracy

- BacktrackSTL is as accurate as offline RobustSTL, better than OneShotSTL.



(a) BacktrackSTL (b) OneShotSTL (c) RobustSTL

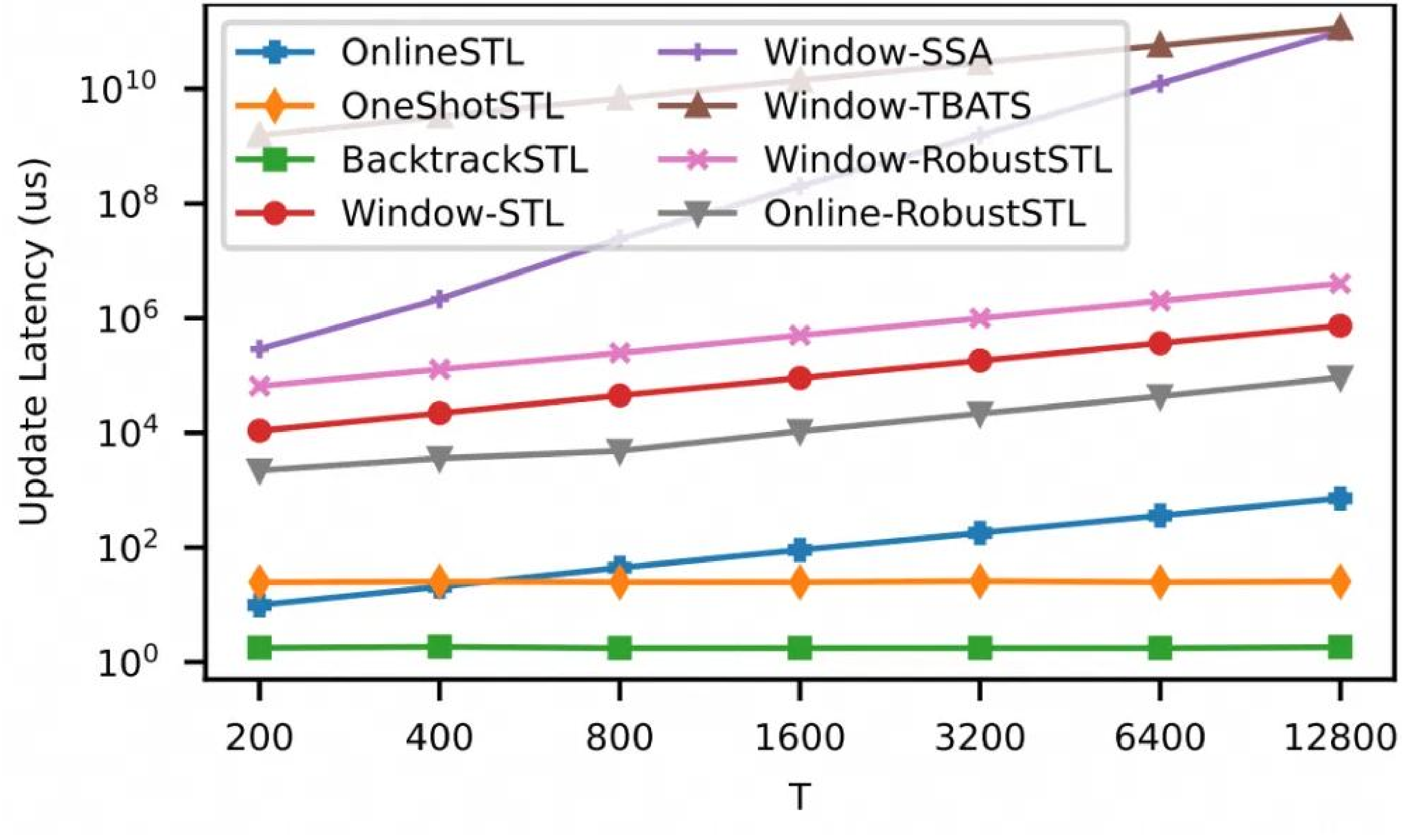
Accuracy

- BacktrackSTL is as accuracy as offline RobustSTL, better than OneShotSTL.

Algorithm	Type	Trend MAE	Seasonality MAE
STL	Offline	0.085	**0.017**
SSA	Offline	0.169	0.152
TBATS	Offline	0.066	0.064
RobustSTL	Offline	**0.010**	0.027
Window-STL	Online	0.165	0.066
Window-SSA	Online	0.444	0.426
Window-TBATS	Online	0.339	0.115
Window-RobustSTL	Online	0.071	0.030
Online-RobustSTL	Online	0.073	0.030
OnlineSTL	Online	0.368	0.303
OneShotSTL	Online	0.150	0.077
BacktrackSTL	Online	**0.012**	**0.023**

Efficiency

- Latency of BacktrackSTL is independent with period length and 15x faster than OneShotSTL.



Conclusion

- In this paper, we introduce BacktrackSTL, a novel seasonal-trend decomposition algorithm with **O(1)** time complexity.
- BacktrackSTL decomposes a value within 1.6us, which is **15x faster** than state-of-the-art online algorithm OneShotSTL.
- BacktrackSTL is robust to trend jumps, seasonality shifts, and outliers. It achieves a **comparable accuracy** to the best offline algorithm RobustSTL.

THANKS FOR YOUR LISTENING

Slides and poster will be shown soon on https://wanghy.pages.dev

Complexity Analysis

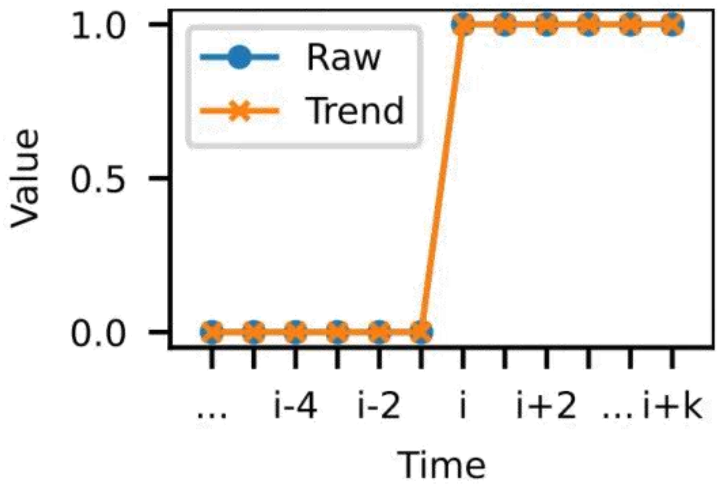
Step	Time Complexity	Frequency	Amortized Complexity
Outlier-resilient smoothing	O(KH)	Always	O(KH) ~ **O(1)**
Non-local seasonal filtering	O(KH)	Always	
Jump detection	O(1)	Always	
Backtrack	O(W+KHL)	<1/W	

Discussion on Jump Detection

- A delayed decision is naturally embedded in the online scenario.
  - BacktrackSTL: Consecutive outlier threshold **(Explicit)**
  - RobustSTL: Regularization parameter **(Implicit)**

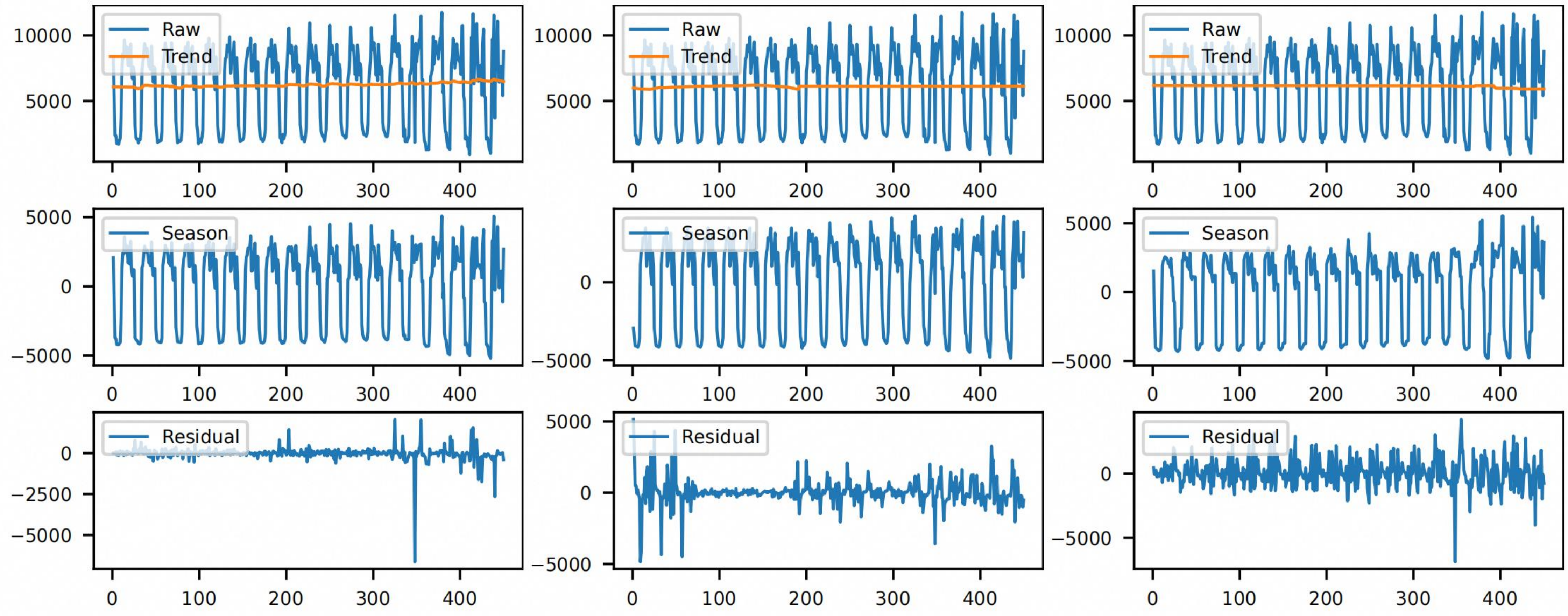
$$\min_{\tau_{1\ldots N}} \sum_{t=T+1}^{N} |(y_t - \tau_t) - (y_{t-T} - \tau_{t-T})| + \lambda_1 \sum_{t=2}^{N} |\tau_t - \tau_{t-1}| + \lambda_2 \sum_{t=3}^{N} |\tau_t - 2\tau_{t-1} + \tau_{t-2}|$$

$$\lambda_1 + 2\lambda_2 < k + 1$$



Accuracy on REAL1

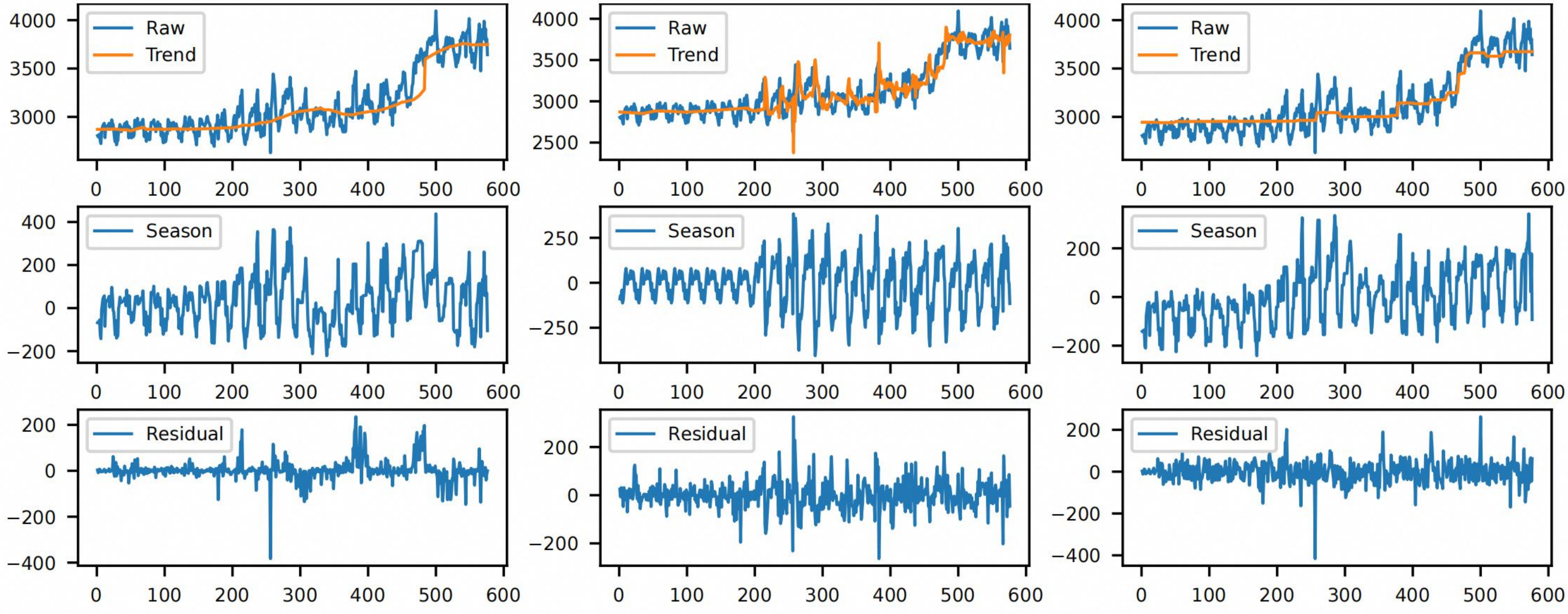
- BacktrackSTL is as accuracy as offline RobustSTL, better than OneShotSTL.



(a) BacktrackSTL on REAL1 (b) OneShotSTL on REAL1 (c) RobustSTL on REAL1

Accuracy on REAL2

- BacktrackSTL is as accuracy as offline RobustSTL, better than OneShotSTL.



(a) BacktrackSTL on REAL2 (b) OneShotSTL on REAL2 (c) RobustSTL on REAL2