Exploring Hierarchical Patterns for Alert Aggregation in Supercomputers

Yuan Yuan[1], Tongqing Zhou[1], Xiuhong Tan[2], Yongqian Sun[3],
Yuqi Li[1,4], Zhixing Li[1], Zhiping Cai[1] and Tiejun Li[1]

1.National University of Defense Technology
2.Changsha University of Science & Technology
3.Nankai University
4.National Supercomputer Center in Tianjin

ISSRE, Oct. 2024

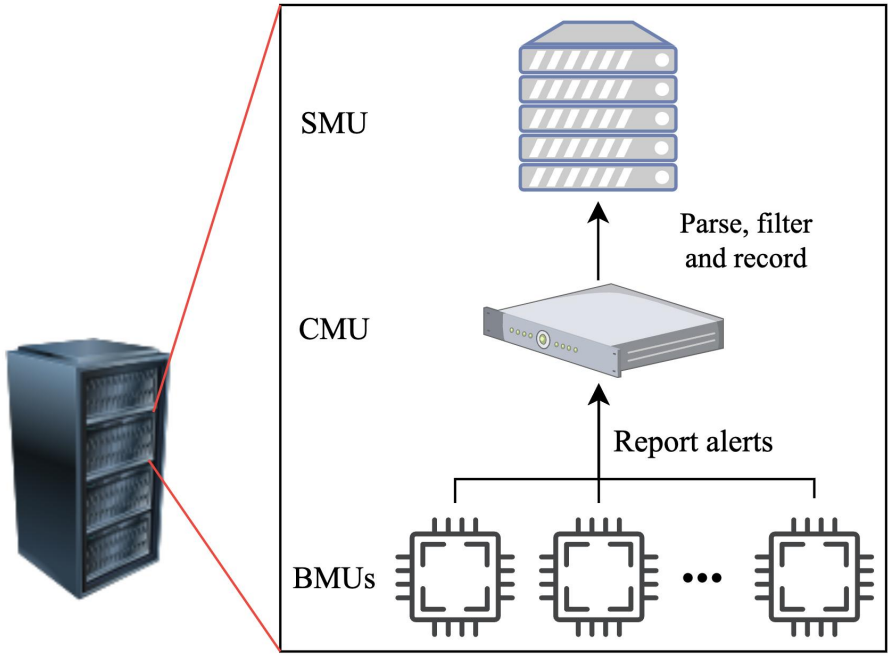
Contents

Contents

Background & Motivation

- Thousands of boards in a supercomputer carry tons of sensors, generating **a huge amount of out-of-band alerts** (i.e., IPMI alerts).
- We adopt a **hierarchical alert reporting.** Even if filtered by blacklists or increased sampling intervals, the system still exposed to **thousands of alerts every day.**

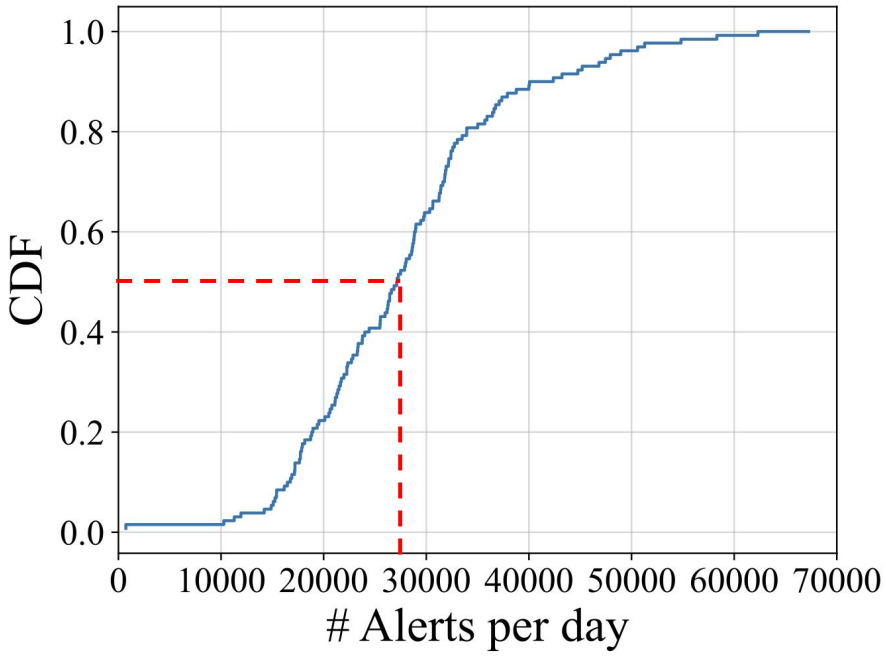
New generation TIANHE

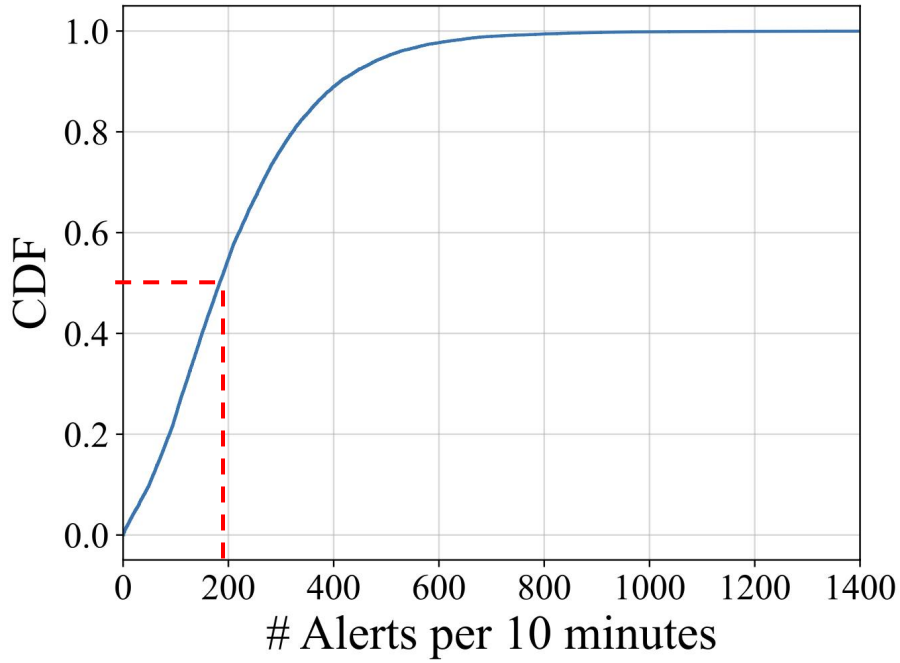


Hierarchical alert reporting

Background & Motivation

- Alert Overload
  - Unlike occasional alert bursts from online services in data centers.
  - **Alert overload** is continuous disturbance of alerts.



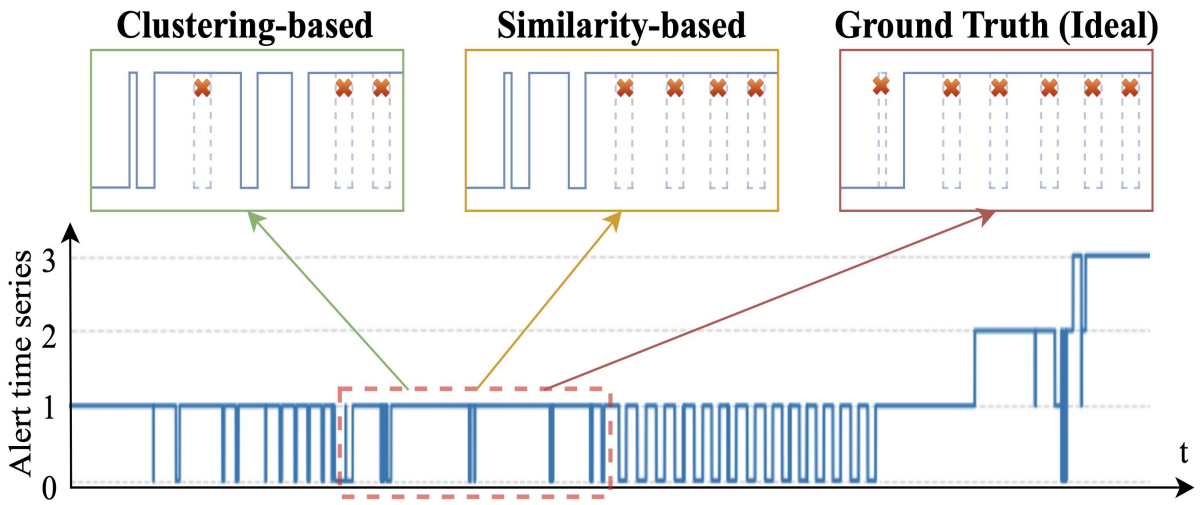


This calls for Automatic Alert Aggregation.

Contents

Challenges

- **Challenge1:** Shape-based aggregation methods do not work.
  - E.g., Clustering-based and Similarity-based methods both need calculating distances, while all the changes in sensor lines are jumps.
- **Challenge2:** Deriving causal relationships from the physical meaning of sensors do not work.
  - Sensor names are very complex and not readable.



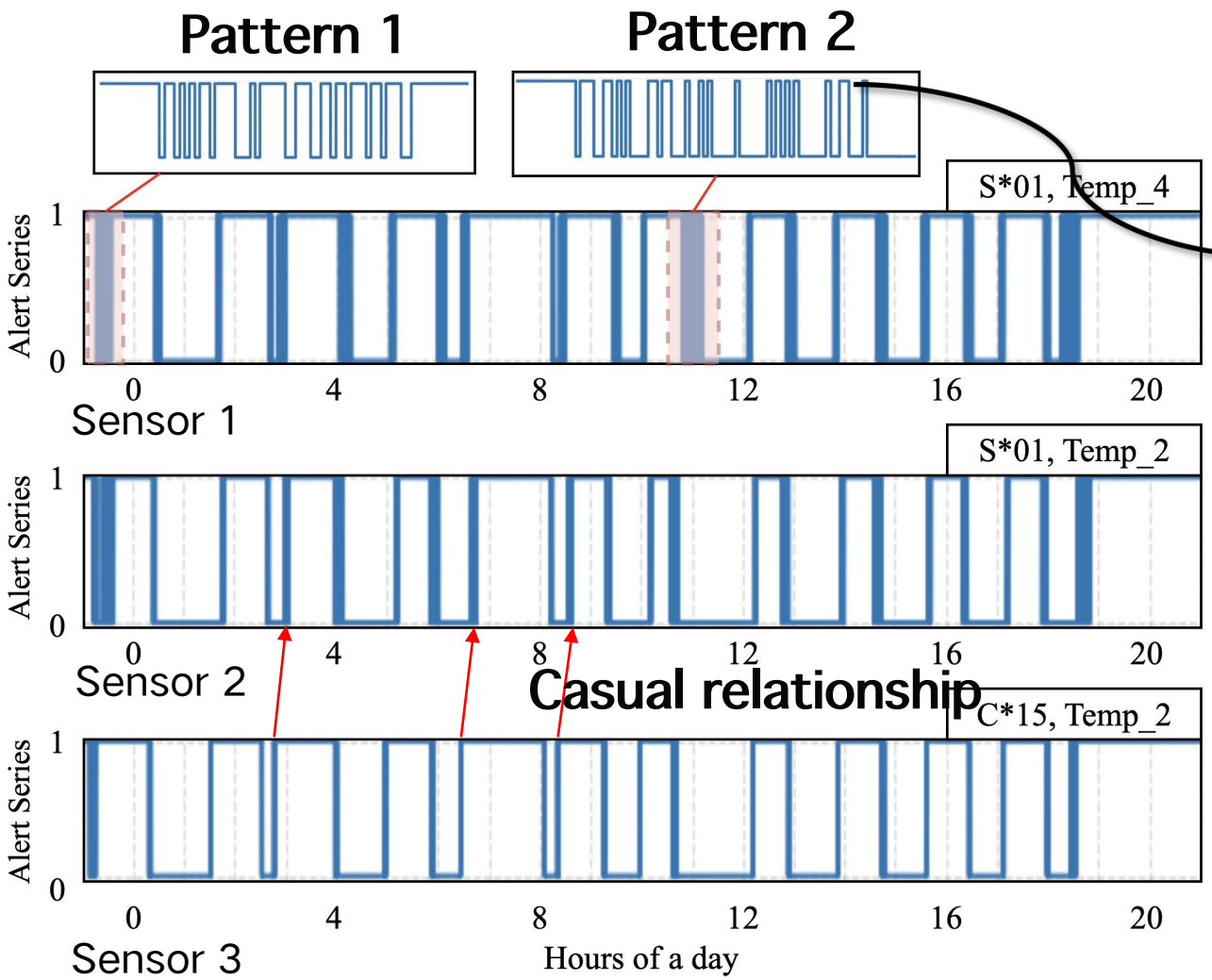
A sensor alert line: jumpy, non-smooth

Sensor ID	Board Name	IP	LF
Temp_3	S*09	192.*.*.*	R3-P*
Volt_1	C*01	192.*.*.*	R3-P*

Sensor names: complex and unreadable

Contents

Observations



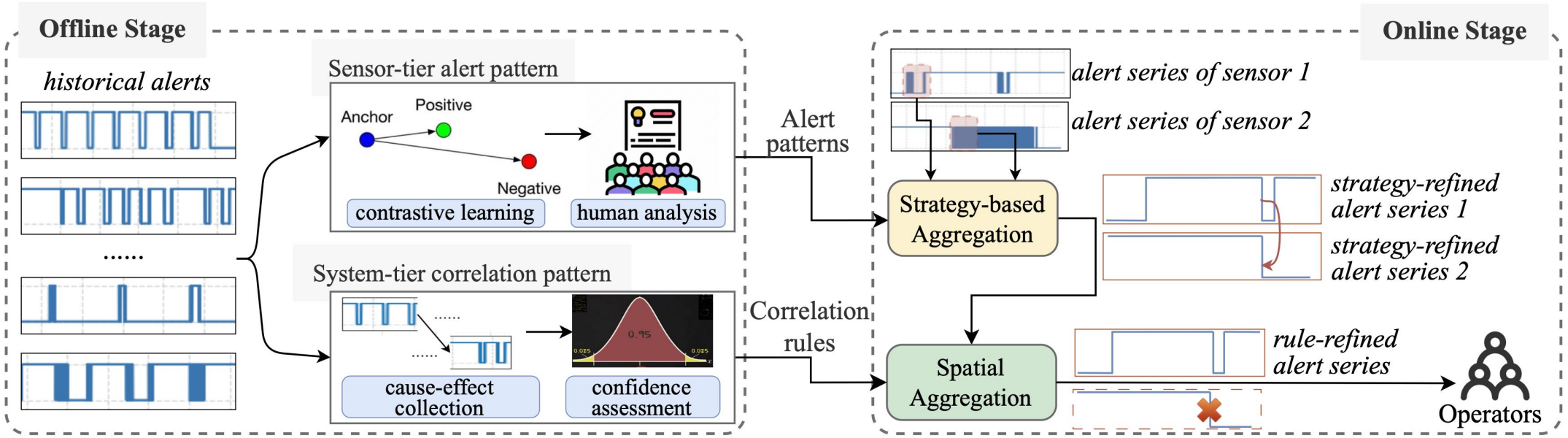
Observation 1: Many frequent bursts of different patterns contribute to the large number of alerts. (redundancy)

Observation 2: These two alert lines show similar trends with a casual relationship. (byproduct)

These motivate us to find out the burst patterns in each sensor, and the causal patterns between sensors.

SuperAgg: Overview

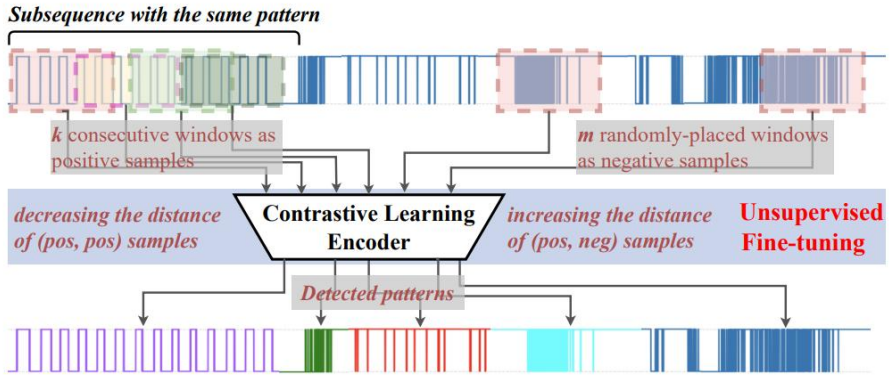
- Offline Stage: learning knowledge Hierarchically from historical alerts
  - Tier1: sensor-tier burst pattern modeling (According to Observation 1)
  - Tier2: system-tier causal pattern modeling (According to Observation 2)
- Online Stage: performing aggregation based on the knowledge
  - Step1: strategy-based aggregation with tier1 patterns
  - Step2: rule-based aggregation with tier2 patterns



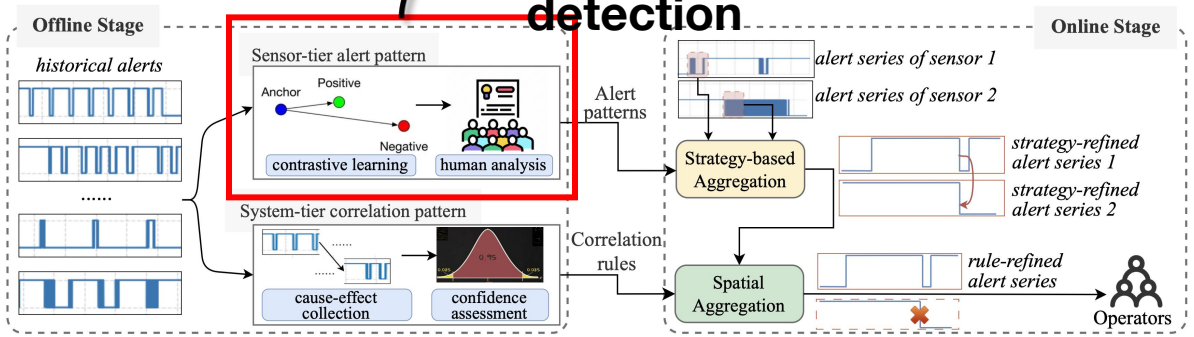
SuperAgg: Offline Pattern Modeling

- **Tier1:** sensor-tier burst pattern modeling

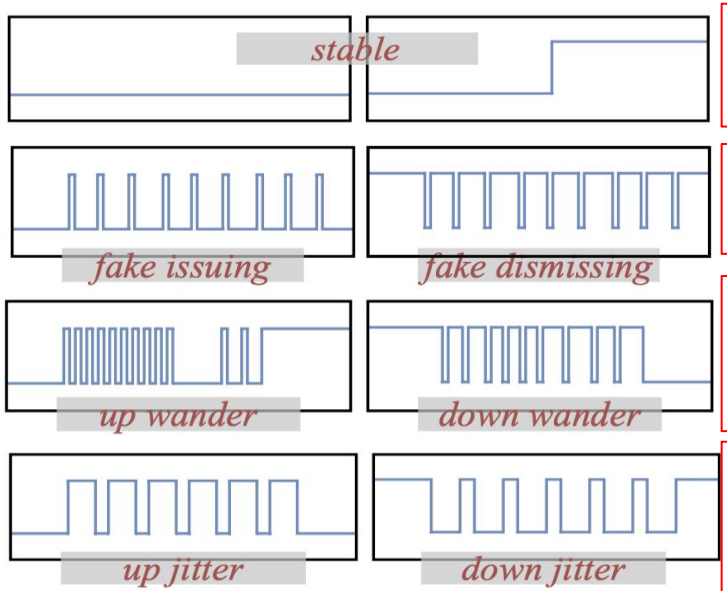
1. Pattern detection by using **contrastive learning.**
   - Time2State , a best SOTA method.



Pattern detection



2. Human-in-the-loop Modeling for **their semantic meaning**
   - Groups 8 patterns into 4 categories
   - It is one-pass step and offline



Stable: The normal and ideal behavior

Fake: The manufacture deficiencies of sensors

Wander: Sensory value fluctuation in critical interval

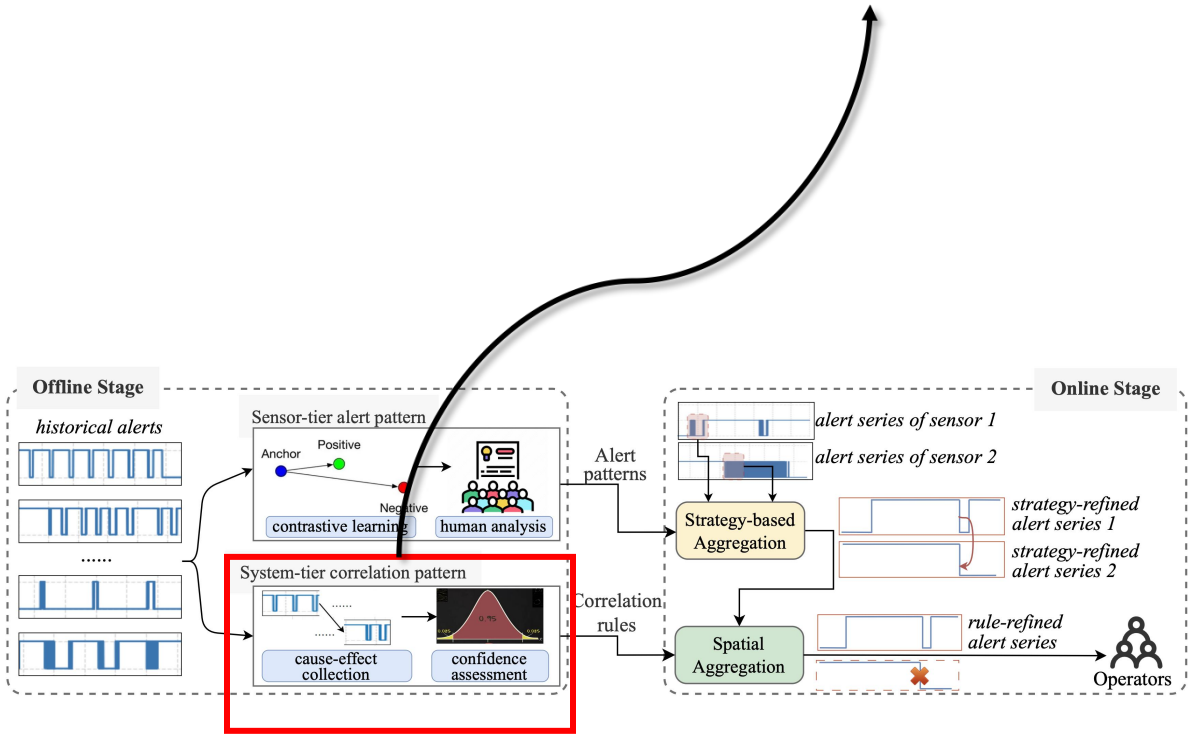
Jitter: Slight sensory value changing with the status of workloads.

Groups 8 patterns into 4 categories by human for different aggregation rules at Online Stage.

SuperAgg: Offline Pattern Modeling

- **Tier2:** system-tier causal pattern modeling

  - Use **a directional Apriori method** to depict the spatial correlation patterns.
  - Generate **primary-and-secondary rules.**

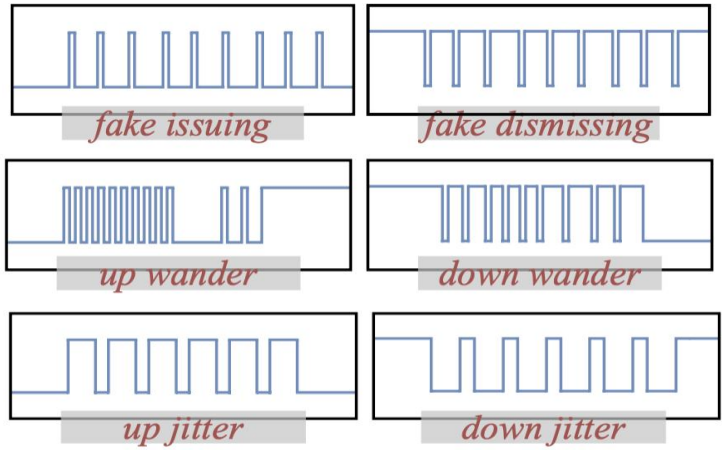


rules	confidence
('33', '41', 'STB_2V5')->('5', '5', 'Temp_3')	0.975
('33', '41', 'STB_2V5')->('6', '6', 'Temp_4')	0.975
('19', '8', 'P5V')->('3', '11', 'Temp_3')	0.971
('19', '8', 'P5V')->('6', '6', 'Temp_4')	0.971
('13', '2', 'NIO_TOP1_Temp')->('13', '2', 'NIO_BOT2_Temp')	0.966
('7', '11', 'STB_2V5')->('3', '11', 'Temp_3')	0.964
('8', '11', 'STB_2V5')->('10', '10', 'Temp_3')	0.963
('8', '11', 'STB_2V5')->('18', '42', 'Temp_1')	0.963
('30', '31', 'NIO_BOT3_Temp')->('30', '31', 'NIO_BOT2_Temp')	0.963
('7', '6', 'STB_2V5')->('9', '9', 'Temp_3')	0.955

- Only report and address the **primary alert**
- **The subsequent alert shall be suppressed**

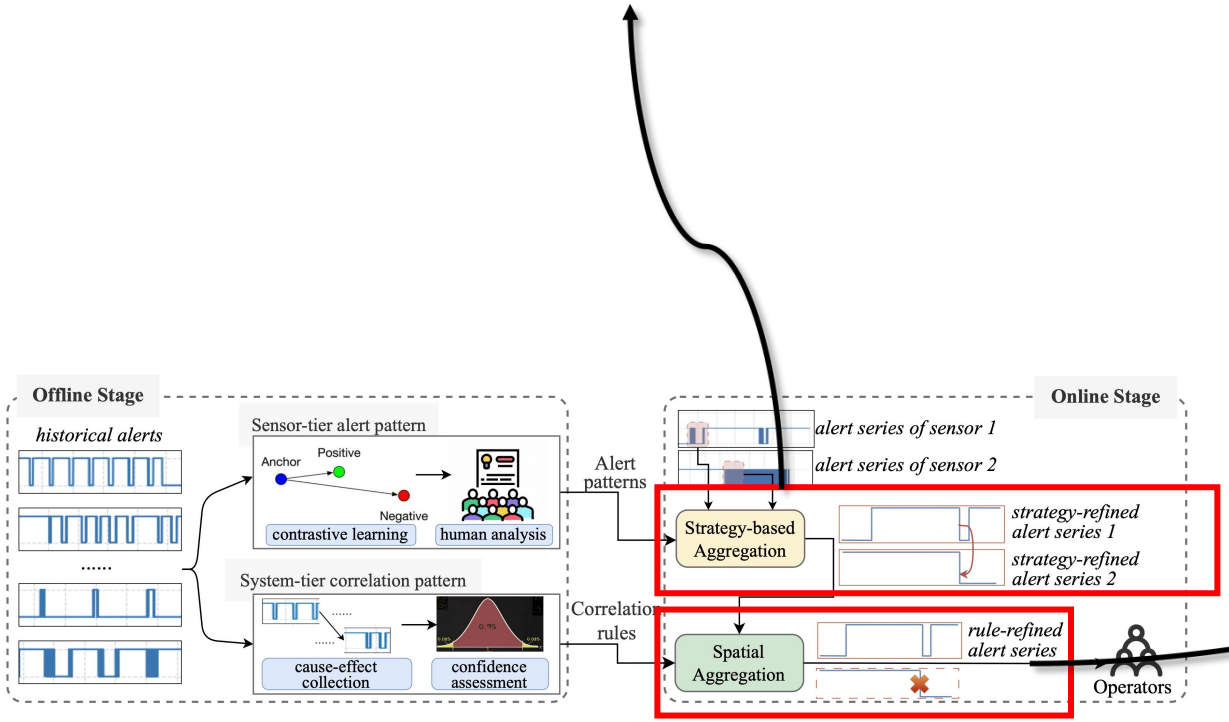
SuperAgg: Online Aggregation

- **Step1**: Strategy-based Aggregation
  - Strategy1: **Silent awaiting** (for **Fake** & **Wander** patterns)
  - Strategy2: **See&suppression** (for **Jitter** patterns)



- **Step2**: Rule-based Aggregation

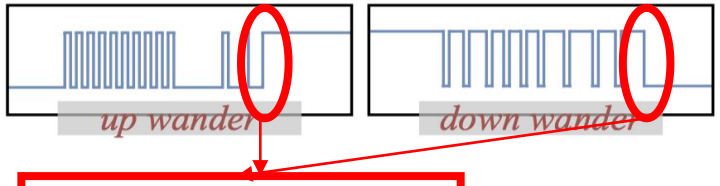
$$Rule_k = [(*, *, AL_i) \rightarrow (*, *, AL_j), 0.8] \in \mathcal{R} \wedge (t_i - t_j) \leq w$$



Contents

Evaluation: Datasets and Metrics

Two datasets:



Datasets	Time span	#Alerts	#Sentinel alerts
A	2023/01/28 ~ 03/31	1552942	607
B	2023/04/01 ~ 06/06	2115815	558

Metrics:

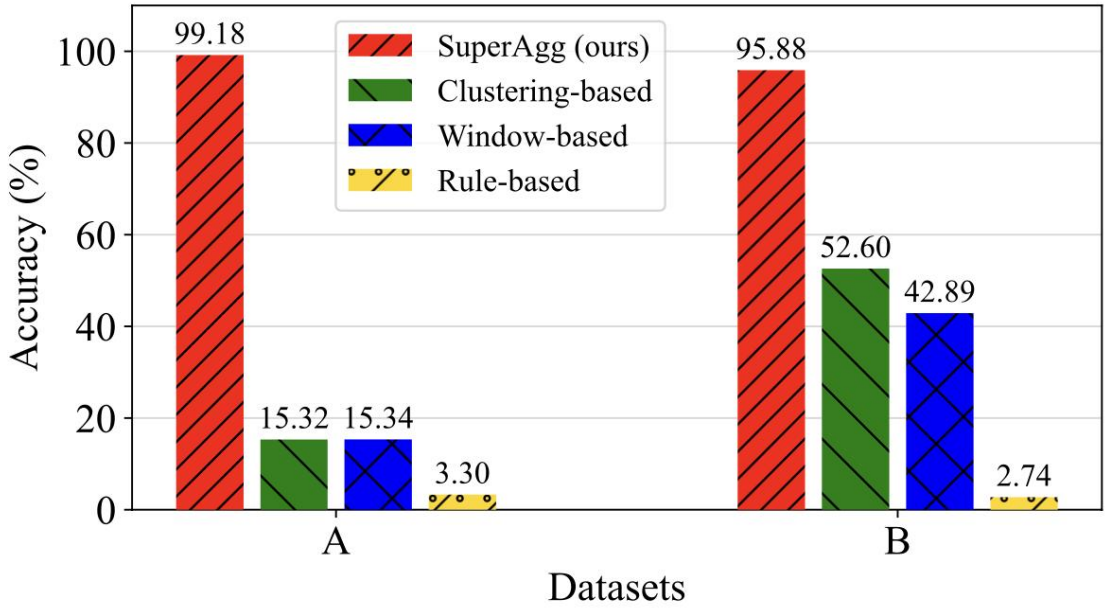
$$\text{Aggregation rate} = \frac{n_{before} - n_{after}}{n_{before}} \times 100\%$$

$$\text{Accuracy} = \frac{n_{retainedSentinels}}{n_{sentinels}} \times 100\%$$

Evaluation: Performance of SuperAgg

Methods	Aggregation Rate (%)	
	Dataset A	Dataset B
Rule-based	97.10	97.74
Clustering-based	97.77	94.81
Window-based	97.87	97.33
SuperAgg (ours)	**99.04**	**98.64**

Improvement of about 0.9% to 3.83%.



At least 83.8% lift on Dataset A, 43.2% on Dataset B.

Implementation on TIANHE

Before SuperAgg, the number of alerts is about 1350 to 9180 per day.

After SuperAgg, the number of alerts is about 57 to 470 per day.

Contents

Conclusion

- This paper is the first work to solve the alert overload problem for supercomputers.
- SuperAgg first detect the burst patterns in each sensor using contrastive learning, then mining the causal patterns in a system-tier.
- SuperAgg has a high aggregation rate and do not miss important alerts.

Thanks for your attention!

Q&A

Email: sunyongqian@nankai.edu.cn