Intelligent Reliability Monitoring and Engineering for Online Service Systems

CHEN, Zhuangbin

Ph.D. Oral Defense
Supervisor: Prof. Michael R. Lyu
30 November, 2022
Online Services are Everywhere

- Web search
- Office apps
- Social network
- Online shopping

And many others…
Service Reliability is Crucial

Service reliability is vital for both service providers and users.

- Revenue loss
- Service issues
- User dissatisfaction
2021 Facebook Outage

State-of-the-art service reliability: 5-6 9s (99.9999% up time)

Facebook service traffic during 2021 outage

**Image from: https://en.wikipedia.org/wiki/2021_Facebook_outage

Three nines left in that year!

$47 billion loss*
Reliability monitoring for online service systems is crucial, but challenging.
Service Reliability is Challenging

Challenge 1: Large scale and complexity


21 Terabytes of Open Source Code Is Now Stored in an Arctic Vault

Other artifacts stored in the archive include manuscripts from the Vatican Library and masterpieces from the National Museum of Norway.
Service Reliability is Challenging

Challenge 2: Fast development iteration

Image from: https://github.com/microsoft/vscode/pulse

38 authors have pushed 185 commits to main and 217 commits to all branches. On main, 337 files have changed and there have been 5,832 additions and 3,562 deletions.

158 Pull requests merged by 37 people
Service Reliability is Challenging

Challenge 3: Complicated service dependencies

A prototype of Google search service

Traditional engineering techniques are often **insufficient**

Intelligent service monitoring is **in need**

- AI Techniques
- Intelligent Service Monitoring
- Big IT Data
Key Qualities of Intelligent Service Monitoring

- Large scale and complexity
- Fast development iteration
- Complicated service dependencies

Good performance: accurate, fast, and high-coverage
Adaptivity and interpretability
Impact scope estimation

21 Terabytes of Open Source Code Is Now Stored in an Arctic Vault

Other artifacts stored in the archive include manuscripts from the Vatican Library and masterpieces from the National Museum of Norway.
Intelligent Service Monitoring

Service usage

Intelligent service monitoring

1. An empirical study on industrial incident management
2. A systematic review on DL-based log anomaly detection
3. Interpretable and adaptive performance anomaly detection
4. Unsupervised and unified alert aggregation

Monitoring data collection

Apps & Services

VM & Containers

Network infrastructure

Logs

Metrics

Alerts/Events

Topology
Thesis Contributions

1. An empirical study on industrial incident management
   Identify the key problems of intelligent service monitoring
   (Chapter 4)
   [FSE '20, AAAI '20, SIGOPS '22]

2. A systematic review on DL-based log anomaly detection
   Help customize and integrate end2end solutions into services
   (Chapter 5)
   [arXiv '21, CSUR '21]

3. Interpretable and adaptive performance anomaly detection
   Accumulate human knowledge for anomaly explanation
   (Chapter 6)
   [ICSE '22, ICSE '23 (in submission)]

4. Unsupervised and unified alert aggregation
   Accelerate failure understanding and impact scoping
   (Chapter 7)
   [ASE '21, ICSE '23 (in submission)]
Outline

- Topic 1: An empirical study on industrial incident management
- Topic 2: Interpretable and adaptive performance anomaly detection
- Topic 3: Unsupervised and unified alert aggregation
- Conclusion and Future work
Outline

Intelligent service monitoring

1. An empirical study on industrial incident management (Chapter 4)
2. A systematic review on DL-based log anomaly detection (Chapter 5)
3. Interpretable and adaptive performance anomaly detection (Chapter 6)
4. Unsupervised and unified alert aggregation (Chapter 7)
Content

o Topic 1: An empirical study on industrial incident management
  ✓ Motivation & methodology
  ✓ Incident characteristics
  ✓ Key challenges of incident management
  ✓ Summary
What is a Service Incident?

Service interruption or performance degradation
- Is or will be affecting user experience
- Can be referred to as *failure*
- Examples
  - Bad HTTP requests
  - Power outages
  - Customer-reported errors
Incident Management Procedure

Incident management procedure
- Incident reporting
- Incident triage
- Incident mitigation
Motivation

- A lack of comprehensive study of incident management
- Understand the key challenges of incident handling
- Identify the unaddressed problems of service monitoring
Methodology

Raw dataset
- Two years of incident tickets at Microsoft

Six core services
- Datacenter Management (DCM)
- Networking
- Storage
- Compute
- Database
- Web Service (WS)

Study approaches
- Incident ticket analysis
- Field studies
- Validation through quantitative experiments

Summary
Writing to a big data storage platform experienced high failure counts.

Diagnosis
Firmware upgrade to a game drive service inadvertently disabled write cache. At the beginning, there was no direct impact on the service because the number of machines getting into bad state was small and the system was built to tolerate such instances. However, as more and more machines were getting upgraded, the overall latency of the service stack was slowly accumulating and at some point got tipped. It took quite some time to detect the incident which unfortunately deteriorated into a critical issue.

An example of incident ticket

The cloud stack of Microsoft Azure
Topic 1: An empirical study on industrial incident management
  ✓ Motivation & Study methodology
  ✓ Incident characteristics
  ✓ Key challenges of incident management
  ✓ Summary
Incident Characteristics

Incident root causes
- Human Errors
- Network Issues
- Deployment Issues
- External Issues
- Capacity Issues
- Others

Distribution of incident root causes

<table>
<thead>
<tr>
<th>Root Cause</th>
<th>Dist.</th>
<th>Root Cause</th>
<th>Dist.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network (Hardware)</td>
<td>22.95%</td>
<td>Human Error (Code Defect)</td>
<td>19.23%</td>
</tr>
<tr>
<td>Network (Connectivity)</td>
<td>2.24%</td>
<td>Human Error (Config.)</td>
<td>7.45%</td>
</tr>
<tr>
<td>Network (Config.)</td>
<td>0.89%</td>
<td>Human Error (Design Flaw)</td>
<td>5.66%</td>
</tr>
<tr>
<td>Network (Other)</td>
<td>4.47%</td>
<td>Human Error (Integration)</td>
<td>2.09%</td>
</tr>
<tr>
<td>Deployment (Upgrade)</td>
<td>5.22%</td>
<td>Human Error (Other)</td>
<td>2.83%</td>
</tr>
<tr>
<td>Deployment (Config.)</td>
<td>3.87%</td>
<td>External Issue (Partner)</td>
<td>2.83%</td>
</tr>
<tr>
<td>Deployment (Other)</td>
<td>1.19%</td>
<td>External Issue (Other)</td>
<td>1.64%</td>
</tr>
<tr>
<td>Capacity Issue</td>
<td>6.56%</td>
<td>Others</td>
<td>10.88%</td>
</tr>
</tbody>
</table>
Incident Characteristics

Incident severity
  - Low + Medium incidents > 90%
  - Critical incidents [0.01%, 0.4%]

Incident fixing time
  - In many cases, the time Critical incidents take is larger than the sum of others

### Distribution of incident severity

<table>
<thead>
<tr>
<th></th>
<th>DCM</th>
<th>Network</th>
<th>Storage</th>
<th>Compute</th>
<th>Database</th>
<th>WS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical</td>
<td>0.01%</td>
<td>0.01%</td>
<td>0.01%</td>
<td>0.31%</td>
<td>0.40%</td>
<td>0.07%</td>
</tr>
<tr>
<td>High</td>
<td>5.48%</td>
<td>1.21%</td>
<td>2.57%</td>
<td>5.27%</td>
<td>4.32%</td>
<td>3.33%</td>
</tr>
<tr>
<td>Medium</td>
<td>86.65%</td>
<td>46.90%</td>
<td>43.32%</td>
<td>74.19%</td>
<td>63.93%</td>
<td>84.52%</td>
</tr>
<tr>
<td>Low</td>
<td>7.86%</td>
<td>51.88%</td>
<td>54.10%</td>
<td>20.23%</td>
<td>31.35%</td>
<td>12.08%</td>
</tr>
</tbody>
</table>

### Distribution of incident fixing time

<table>
<thead>
<tr>
<th></th>
<th>DCM</th>
<th>Network</th>
<th>Storage</th>
<th>Compute</th>
<th>Database</th>
<th>WS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical</td>
<td>38.33x</td>
<td>8.46x</td>
<td>10.06x</td>
<td>142.05x</td>
<td>209.97x</td>
<td>286.6x</td>
</tr>
<tr>
<td>High</td>
<td>19.25x</td>
<td>3.18x</td>
<td>2.52x</td>
<td>2.56x</td>
<td>5.75x</td>
<td>3.56x</td>
</tr>
<tr>
<td>Medium</td>
<td>1x</td>
<td>9.8x</td>
<td>7.09x</td>
<td>2.95x</td>
<td>25.28x</td>
<td>12.93x</td>
</tr>
<tr>
<td>Low</td>
<td>3.01x</td>
<td>5.49x</td>
<td>1.09x</td>
<td>11.65x</td>
<td>2.41x</td>
<td>144.79x</td>
</tr>
</tbody>
</table>
Content

- Topic 1: An empirical study on industrial incident management
  - Motivation & Study methodology
  - Incident characteristics
  - Key challenges of incident management
  - Summary
## Key Challenges of Incident Management

**Challenge 1: Resource health assessment**

- Problem detection based on various signals (metrics, logs, etc.)
- Hard-to-understand problems with complex and changing patterns
Key Challenges of Incident Management

Challenge 1: Resource health assessment
- Problem detection based on various signals (metrics, logs, etc.)
- Hard-to-understand problems with complex and changing patterns

Accurate, adaptive, and interpretable anomaly detection alleviates flooding alarms and gray failures [Topic 2]

Key Challenges of Incident Management

Challenge 2: Resource dependency discovery
- Services rely on each other (microservices)
- Incomplete, outdated, and human-dependent
Key Challenges of Incident Management

Challenge 2: Resource dependency discovery

- Services rely on each other (microservices)
- Incomplete, outdated, and human-dependent

Identifying related problems facilitates failure impact estimation and duplicate effort saving [Topic 3]
Understanding the Key Challenges

Challenge 1: Resource health assessment
- System fault tolerance
- Monitor design and distribution
- ...

Challenge 2: Resource dependency discovery
- Software system modularity
- Physical infrastructure virtualization
- Dynamic deployment
- Load balancing
- ...

A typical cloud computing architecture

Incident ID
Resolved
Critical

Disk firmware update disabled disk cache
Service: Storage
Datacenter: DC #4
# of impacted requests: ~100,000
# of impacted accounts: ~10,000

Summary
Monitor has detected multiple VMs and web applications unavailable.

Diagnosis
Some operations of Cloud Resource Management (CRM) service suffered from a high error rate. Engineering team found the frontend web service was in a loop of crash and rebal. This resulted in customer requests being held for an extended period of time in web server request queue, leading to slow responses and request timeouts. More than five other services suffered from different failures such as login failures, request timeout errors, etc. The cascading effects and implicit service dependencies made the engineering team hard to know and notify all impacted service teams, especially during busy bug fixing time. Therefore, many impacted services received failure reports and diagnosed their services independently. Particularly, an IT Management Software (ITMS) service attributed the failures to DNS service due to the direct dependency. However, the DNS service was managed by the CRM service (the true root cause), which took ITMS team some time to figure out.
Content

- **Topic 1: An empirical study on industrial incident management**
  - ✓ Motivation & Study methodology
  - ✓ Incident characteristics
  - ✓ Key challenges of incident management
  - ✓ Summary
Summary of Topic 1

- A comprehensive study of industrial incident management
- The general management procedure of incidents and their characteristics
- Study the key challenges of incident handling and the underlying reasons
- Findings motivate the studies in Topic 2 and Topic 3
Outline

Intelligent service monitoring

1. An empirical study on industrial incident management (Chapter 4)

2. A systematic review on DL-based log anomaly detection (Chapter 5)

3. Interpretable and adaptive performance anomaly detection (Chapter 6)

4. Unsupervised and unified alert aggregation (Chapter 7)
Topic 2: Interpretable and adaptive performance anomaly detection

**Motivation**

- Anomaly detection based on pattern sketching
- Evaluation
- Summary
Performance Anomaly Detection

Performance anomalies
  - Slow service response
  - High temperature
  - ...

Service performance is monitored with metrics
  - Request latency
  - Request success rate
  - Traffic volume
  - ...

An anomaly is an observation or a sequence of observations which deviates remarkably from the general distribution of data [1].

Why Yet Another Detection Algorithm?

Indeed, many existing unsupervised approaches

- Forecasting-based: LSTM
- Reconstruction-based: Donut, LSTM-VAE
- Probabilistic: LODA, DAGMM, Extreme Value Theory
- Tree-based: Isolation Forest
- Others: SR-CNN, ...

In production, we need

- **Interpretability**: gain engineers’ trust, accelerate failure understanding
- **Online adaptability**: accommodate unseen patterns
- **Human knowledge reusage**: valuable company asset
Motivating Observations

Key observations

- Metric time series tends to develop individual and stable patterns
  - A metric pattern: repeated similar subsequences
  - Similar observations have been made [1-3]

- Similar anomalies incur similar anomalous patterns on the metric time series [4]

- Find metric patterns
- Distinguish the anomalous patterns from the normal ones
- Adapt to unseen patterns

Motivating Observations

Anomaly detection strategy – **Pattern Sketching**
- When a service runs normally, it produces normal patterns
- If a new pattern deviates substantially from the normal ones, it could be abnormal

**Interpretability**
- If a known abnormal patterns is detected, we know what performance anomalies have happened
Content

- Topic 2: Interpretable and adaptive performance anomaly detection
  - Motivation
  - Anomaly detection based on pattern sketching
  - Evaluation
  - Summary
ADSketch Overview

1. Metric pattern discovery (offline anomaly detection)

2. Online anomaly detection

3. Adaptive pattern learning
The Smallest Pair-Wise (SPW) Distance

- A subsequence: a continuous part of a metric time series
- The SPW distance of a subsequence: its smallest distance to other subsequences
- If a subsequence has a large SPW distance, it is likely an anomaly

Brute-force searching is not scalable
- STAMP [1] is faster by orders of magnitude
  - Fast Fourier Transform (FFT)

Metric Pattern Discovery

- Algorithm inputs
  - 1. Anomaly-free time series
  - 2. Time series for anomaly detection

- Algorithm outputs
  - Anomalies
  - Normal and abnormal patterns

Algorithm 1: Performance Anomaly Pattern Discovery

**Input:** $T_n, T_a, m,$ and $p$

**Output:** Two disjoint sets of $\mathcal{P}_n$ and $\mathcal{P}_a$

1. $I_{nn}, S_{nn} \leftarrow \text{STAMP}(T_n, T_a, m)$
2. $I_{na}, S_{na} \leftarrow \text{STAMP}(T_n, T_a, m)$
3. $G \leftarrow \text{ConnectedSubgraphs}(I_{nn} + I_{na}, S_{na}, p)$
4. $N_i \leftarrow \text{IsolatedNodes}(G)$
5. $\mu_G \leftarrow \text{GraphWiseMean}(G)$
6. $C \leftarrow \text{AffinityPropagation}(\mu_G)$
7. $\mu_C \leftarrow \text{ClusterWiseMean}(C)$
8. $\mathcal{P}_n \leftarrow \text{EmptyArray}, \mathcal{P}_a \leftarrow \text{EmptyArray}$
9. **for** each idx in 1: Size(C) **do**
   - // $C[\text{idx}]$: all subsequences in the cluster
   - if $C[\text{idx}] < N_i$ then
     - $\mathcal{P}_a \leftarrow \text{Append } \mathcal{P}_a \text{ with idx}$
   - else
     - $\mathcal{P}_n \leftarrow \text{Append } \mathcal{P}_n \text{ with idx}$
   - **end**
10. **end**

- Break due to percentile threshold unfulfillment
- Isolated subgraphs, also the anomaly candidates
- Apply Affinity Propagation to the mean of each subgraph
- The mean of each cluster
- Metric patterns, and is the only abnormal pattern
ADSketch Overview

1. Metric pattern discovery (offline anomaly detection)

2. Online anomaly detection

3. Adaptive pattern learning
Online Anomaly Detection

- Algorithm inputs
  - Streaming time series for anomaly detection

- Algorithm outputs
  - Anomalies in the time series

Algorithm 2: Performance Anomaly Detection

<table>
<thead>
<tr>
<th>Input: $t$, $P_a$, and $\mu_C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: Anomaly detection result for $t$</td>
</tr>
</tbody>
</table>

1. $D_t \leftarrow$ PairWiseDistance($t$, $\mu_C$)
2. $idx \leftarrow$ MinIndex($D_t$)
3. if $idx \in P_a$ then
   4. return True
5. else
6. return False
7. end

Stream metric time series

<table>
<thead>
<tr>
<th>Metric patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomalous pattern</td>
</tr>
</tbody>
</table>

Prediction results

- ✔️
- ✔️
- ✔️
- ❌
ADSketch Overview

1. Metric pattern discovery (offline anomaly detection)

2. Online anomaly detection

3. Adaptive pattern learning
Adaptive Pattern Learning

- Algorithm inputs
  - Streaming time series for anomaly detection

- Algorithm outputs
  - Anomalies
  - Updated metric patterns

Stream metric time series

A new anomaly pattern!

Metric patterns

The mean

- close enough to ?
  - Yes $\Rightarrow$ Update the pattern
  - No $\Rightarrow$ Create a new anomalous pattern

Graphic representation of algorithm steps:

1. $t = \text{PairWiseDistance}(t, \mu_C)$
2. $idx = \text{MinIndex}(D_t)$
3. $\mu' = (\mu_C[idx] \times S_C[idx] + t) / (S_C[idx] + 1)$
4. $d_w = \text{Distance}(\mu_C[idx], \mu') + R_C[idx]$
5. $d_t = \text{Distance}(t, \mu')$
6. $d = \max(d_t, d_w)$
7. $d_a, d_u = \max(R_C[P_a]), \max(R_C[P_a])$
8. If $idx \in P_a$ then $d = d_a$ else $d = d_u$ end
9. If $d \leq d_a$ then 
   - add $t$ to the most similar cluster
   - $\mu_C[idx], S_C[idx], R_C[idx] = \mu', S_C[idx] + 1, d'$
   - If $S_C[idx] > \max(S_C[P_a])$ and $idx$ is a new cluster
     - $P_a \leftarrow \text{Append } P_a \text{ with } idx$
     - $P_a \leftarrow \text{Remove } idx \text{ from } P_a$
   - Else
     - $d = \max(d, d')$ // $d$ will be assigned to $d_a$
     - or $d_u$ accordingly

10. End

11. // create a new anomalous cluster for $t$
12. $P_a \leftarrow \text{Append } P_a \text{ with } \text{Length}(\mu_C) + 1$
13. $\mu_C \leftarrow \text{Append } \mu_C \text{ with } t$
14. $R_C \leftarrow \text{Append } R_C \text{ with } 0$
15. $S_C \leftarrow \text{Append } S_C \text{ with } 1$

The worst case for pattern updates
Complexity Analysis

- **Time complexity**
  - ✓ The closest pair searching: \( O(n^2) \)
  - ✓ Affine propagation algorithm: \( O(|C|^2) \), \(|C|\) is the number of clusters, which is small
  - ✓ Online anomaly detection and pattern updating: \( O(n) \)
  - ✓ Overall: \( O(n^2) \)
  - ✓ Easily parallelizable
  - ✓ Ultra-fast approximation is attainable

- **Space complexity**
  - ✓ The indexes of metric patterns: \( O(|C|) \)
  - ✓ The storage of metric patterns: \( O(m \times |C|) \), \( m \) is the length of subsequences
  - ✓ Our design makes it trivial
Content

- **Topic 2: Interpretable and adaptive performance anomaly detection**
  - ✓ Motivation
  - ✓ Anomaly detection based on pattern sketching
  - ✓ Evaluation
  - ✓ Summary
Evaluation Questions

- RQ1: How effective is ADSketch’s offline anomaly detection?
- RQ2: How effective is ADSketch’s online anomaly detection?
- RQ3: How effective is ADSketch’s adaptive pattern learning?
Experiment Settings

- Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Curves</th>
<th>#Points</th>
<th>Anomaly Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yahoo</td>
<td>67</td>
<td>94,866</td>
<td>1.8%</td>
</tr>
<tr>
<td>AIOps18</td>
<td>58</td>
<td>5,922,913</td>
<td>2.26%</td>
</tr>
<tr>
<td>Industry</td>
<td>436</td>
<td>4,394,880</td>
<td>1.07%</td>
</tr>
</tbody>
</table>

- Evaluation Metrics

\[
\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}, \quad F1 \text{ score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\]
Experimental Results

- Offline anomaly detection
  - 2.1%-54% improvement in Yahoo
  - 26%-86% improvement in AIOps18
  - 17%-70% improvement in Industry

<table>
<thead>
<tr>
<th>Method</th>
<th>Yahoo precision</th>
<th>Yahoo recall</th>
<th>Yahoo F1 score</th>
<th>AIOps18 precision</th>
<th>AIOps18 recall</th>
<th>AIOps18 F1 score</th>
<th>Industry precision</th>
<th>Industry recall</th>
<th>Industry F1 score</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>0.598</td>
<td>0.706</td>
<td>0.530</td>
<td>0.499</td>
<td>0.531</td>
<td>0.518</td>
<td>0.704</td>
<td>0.656</td>
<td>0.632</td>
</tr>
<tr>
<td>LSTM-VAE</td>
<td>0.622</td>
<td>0.634</td>
<td>0.484</td>
<td>0.510</td>
<td>0.625</td>
<td>0.537</td>
<td>0.717</td>
<td>0.639</td>
<td>0.622</td>
</tr>
<tr>
<td>Donut</td>
<td>0.530</td>
<td>0.658</td>
<td>0.524</td>
<td>0.405</td>
<td>0.527</td>
<td>0.382</td>
<td>0.693</td>
<td>0.628</td>
<td>0.604</td>
</tr>
<tr>
<td>LODA</td>
<td>0.754</td>
<td>0.583</td>
<td>0.428</td>
<td>0.553</td>
<td>0.429</td>
<td>0.401</td>
<td>0.583</td>
<td>0.498</td>
<td>0.529</td>
</tr>
<tr>
<td>iForest</td>
<td>0.713</td>
<td>0.597</td>
<td>0.437</td>
<td>0.555</td>
<td>0.439</td>
<td>0.413</td>
<td>0.616</td>
<td>0.567</td>
<td>0.538</td>
</tr>
<tr>
<td>DAGMM</td>
<td>0.643</td>
<td>0.517</td>
<td>0.401</td>
<td>0.590</td>
<td>0.477</td>
<td>0.461</td>
<td>0.597</td>
<td>0.542</td>
<td>0.530</td>
</tr>
<tr>
<td>SR-CNN</td>
<td>0.433</td>
<td>0.618</td>
<td>0.307</td>
<td>0.424</td>
<td>0.387</td>
<td>0.363</td>
<td>0.519</td>
<td>0.471</td>
<td>0.434</td>
</tr>
<tr>
<td>ADSketch</td>
<td>0.511</td>
<td>0.673</td>
<td>0.541</td>
<td>0.744</td>
<td>0.670</td>
<td>0.677</td>
<td>0.811</td>
<td>0.813</td>
<td>0.740</td>
</tr>
</tbody>
</table>
Experimental Results

- Online anomaly detection
  - 24%-65% improvement in AIOps18
  - 0.8%-48% improvement in Industry

<table>
<thead>
<tr>
<th>Method</th>
<th>AIOps18 prec.</th>
<th>AIOps18 rec.</th>
<th>AIOps18 F1</th>
<th>Industry prec.</th>
<th>Industry rec.</th>
<th>Industry F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>0.425</td>
<td>0.462</td>
<td>0.408</td>
<td>0.612</td>
<td>0.606</td>
<td>0.592</td>
</tr>
<tr>
<td>LSTM-VAE</td>
<td>0.336</td>
<td>0.521</td>
<td>0.389</td>
<td>0.624</td>
<td>0.598</td>
<td>0.601</td>
</tr>
<tr>
<td>Donut</td>
<td>0.431</td>
<td>0.326</td>
<td>0.376</td>
<td>0.662</td>
<td>0.581</td>
<td>0.590</td>
</tr>
<tr>
<td>LODA</td>
<td>0.407</td>
<td>0.397</td>
<td>0.355</td>
<td>0.653</td>
<td>0.526</td>
<td>0.503</td>
</tr>
<tr>
<td>iForest</td>
<td>0.397</td>
<td>0.334</td>
<td>0.322</td>
<td>0.576</td>
<td>0.507</td>
<td>0.487</td>
</tr>
<tr>
<td>DAGMM</td>
<td>0.392</td>
<td>0.367</td>
<td>0.378</td>
<td>0.557</td>
<td>0.538</td>
<td>0.502</td>
</tr>
<tr>
<td>SR-CNN</td>
<td>0.329</td>
<td>0.288</td>
<td>0.307</td>
<td>0.438</td>
<td>0.422</td>
<td>0.410</td>
</tr>
<tr>
<td>ADSketch</td>
<td><strong>0.543</strong></td>
<td><strong>0.575</strong></td>
<td><strong>0.507</strong></td>
<td><strong>0.705</strong></td>
<td><strong>0.603</strong></td>
<td><strong>0.606</strong></td>
</tr>
</tbody>
</table>
Experimental Results

- Adaptive anomaly detection
  - 35%-42% improvement in AIOps18
  - 52%-83% improvement in Industry

<table>
<thead>
<tr>
<th>Method</th>
<th>AIOps18 prec.</th>
<th>AIOps18 rec.</th>
<th>AIOps18 F1</th>
<th>Industry prec.</th>
<th>Industry rec.</th>
<th>Industry F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>LODA</td>
<td>0.424</td>
<td>0.405</td>
<td>0.387</td>
<td>0.623</td>
<td>0.512</td>
<td>0.548</td>
</tr>
<tr>
<td>EVT</td>
<td>0.455</td>
<td>0.528</td>
<td>0.406</td>
<td>0.710</td>
<td>0.612</td>
<td>0.458</td>
</tr>
<tr>
<td>ADSketch</td>
<td>0.594</td>
<td>0.557</td>
<td>0.548</td>
<td>0.882</td>
<td>0.856</td>
<td>0.832</td>
</tr>
</tbody>
</table>
**Industrial Deployment**

ADSketch has been deployed in Huawei Cloud

- Serve tens of thousands of service instances and devices
- The accuracy of anomaly detection has been substantially improved
- Being integrated into the anomaly detection service for internal users

---

[Image: https://www.huaweicloud.com/lab/cnl/paper_anomaly_detection.html]
Content

- **Topic 2: Interpretable and adaptive performance anomaly detection**
  - ✔ Motivation
  - ✔ Anomaly detection based on pattern sketching
  - ✔ Evaluation
  - ✔ Summary
Summary of Topic 2

- **ADSketch**: A performance anomaly detector based on pattern sketching
  - An explicit metric pattern discovery algorithm
  - An adaptive pattern learning algorithm
  - A labeling scheme to improve interpretability and reuse human knowledge

- ADSketch has been deployed in production and performs well
Outline

Intelligent service monitoring

1. An empirical study on industrial incident management (Chapter 4)
2. A systematic review on DL-based log anomaly detection (Chapter 5)
3. Interpretable and adaptive performance anomaly detection (Chapter 6)
4. Unsupervised and unified alert aggregation (Chapter 7)
Content

- Topic 3: Unsupervised and unified alert aggregation
  - ✓ Motivation
  - ✓ Graph representation learning for alert aggregation
  - ✓ Evaluation
  - ✓ Summary
Alerting in Online Services

- Alerting gives timely awareness to problems in cloud applications
- Monitors render an alert upon alerting policy violation
  - E.g., Specify the values of HTTP response latency that require user responses

Alert title: The HTTP response latency is higher than 2s for at least 5m.

Alert format
Alert ID, Alert type, Alert title, Alert time, Severity, Component, etc.

Setting alert rules in Microsoft Azure

Flooding Alerts

Incidents often come with many alerts
- Complex service dependencies, i.e., cascading effect
- Conservative alerting policies

Pain points of site reliability engineers
- Duplicate engineering efforts
- Delayed root cause analysis
Alert Aggregation

Group alerts associated with the same failure

✓ Estimate failure impact scope
✓ Save duplicate engineering effort

A failure happened to service A

Failure propagation

Failure-impact graph (the circled area)
Challenges

- Background noise
- Little textual similarity
  - “Traffic burst seen in Nginx node” and “Traffic burst seen in LVS node”
  - “Virtual machine is in abnormal state” and “OSPF protocol state change”
- Lack of labeled data
- Incomplete failure-impact graph based on alerts
  - Alerting policies not triggered
  - Fault tolerance bears anomalies
Incorporating Metric Information

- Metrics characterize failure impact in a more fine-grained way

Diagram showing metric similarity and a graph with nodes and edges.
Content

- Topic 3: Unsupervised and unified alert aggregation
  - ✓ Motivation
  - ✓ Graph representation learning for alert aggregation
  - ✓ Evaluation
  - ✓ Summary
Girdle Overview

1. Service failure detection
2. Failure-impact graph completion
3. Graph representation learning
4. Online alert aggregation
Service Failure Detection

- Detect historical failures for alert correlation learning
- Flooding alerts (check the no. of alerts/min)
- Extreme Value Theory (EVT)
  - No hand-set thresholds
  - No assumption on data distribution
Failure-impact Graph Completion

- Identify alerts triggered by the common failure

- Community detection
  - Identify similar node sets in a graph
  - The key is the design of two nodes’ similarity
    - Alert set similarity (Jaccard index)
    - Metric similarity (Dynamic time warping)

- Preliminary correlations between alerts

Dynamic time warping*

Deal with possible clock non-sync between nodes during metric collection
Graph Representation Learning

- Learn more significant correlations between alerts from historical failures

- Existing work combines different features by a simple weighted sum

- Graph representation learning
  - Learn a feature vector $v$ for each unique type of alert
  - Unify the temporal and topological correlations of alerts
Online Alert Aggregation

- Quickly aggregate alerts when failures happen in production environment
- Two alerts $i$ and $j$ will be grouped if their similarity score is large

$$\text{sim}(i, j) = \mathcal{T}(i, j) \times \mathcal{H}(i, j)$$

<table>
<thead>
<tr>
<th>Historical closeness</th>
<th>Topological rescaling</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{H}(i, j) = \frac{v_i \cdot v_j}{</td>
<td></td>
</tr>
</tbody>
</table>
0 Topic 3: Unsupervised and unified alert aggregation
  ✔ Motivation
  ✔ Graph representation learning for alert aggregation
  ✔ Evaluation
  ✔ Summary
Dataset

- Alerts
  - Networking service of Huawei Cloud
  - Alerts are reported by various devices and virtual network function (VNF) instances

- Metrics
  - CPU usage
  - Round trip delay
  - Port in-bound/out-bound traffic rate
  - Package receiving/sending rate
  - Package receiving/sending error rate

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Training period</th>
<th>Testing period</th>
<th>#alerts</th>
<th>#failures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset1</td>
<td>2020 May - July</td>
<td>2020 Aug.</td>
<td>~18k/~8k</td>
<td>105/46</td>
</tr>
<tr>
<td>Dataset2</td>
<td>2020 May - Aug.</td>
<td>2020 Sept.</td>
<td>~26k/~10k</td>
<td>151/52</td>
</tr>
<tr>
<td>Dataset3</td>
<td>2020 May - Sept.</td>
<td>2020 Oct.</td>
<td>~36k/~8k</td>
<td>203/38</td>
</tr>
</tbody>
</table>
Evaluation Metrics

- Service failure detection (binary classification)
  - Precision, Recall, and F1 score
    \[
    \text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}, \quad \text{F1 score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
    \]

- Alert aggregation (clustering)
  - Normalized Mutual Information (NMI) in $[0, 1]$ (the larger the better)
    \[
    \text{NMI}(Y, C) = \frac{2 \times I(Y; C)}{H(Y) + H(C)}
    \]
    \[
    Y = \text{class labels} \quad C = \text{cluster labels} \quad H(\cdot) = \text{Entropy} \quad I(Y; C) = \text{Mutual info b/w } Y \text{ and } C
    \]
Service Failure Detection

- Girdle outperforms simple thresholding by 8.9%-24.7%

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Metric</th>
<th>Threshholding</th>
<th>Girdle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset1</td>
<td>Precision</td>
<td>0.711</td>
<td>0.917</td>
</tr>
<tr>
<td></td>
<td>Recall</td>
<td>0.913</td>
<td>0.957</td>
</tr>
<tr>
<td></td>
<td>F1 Score</td>
<td>0.799</td>
<td>0.937</td>
</tr>
<tr>
<td>Dataset2</td>
<td>Precision</td>
<td>0.831</td>
<td>0.944</td>
</tr>
<tr>
<td></td>
<td>Recall</td>
<td>0.942</td>
<td>0.981</td>
</tr>
<tr>
<td></td>
<td>F1 Score</td>
<td>0.883</td>
<td>0.962</td>
</tr>
<tr>
<td>Dataset3</td>
<td>Precision</td>
<td>0.648</td>
<td>0.925</td>
</tr>
<tr>
<td></td>
<td>Recall</td>
<td>0.921</td>
<td>0.974</td>
</tr>
<tr>
<td></td>
<td>F1 Score</td>
<td>0.761</td>
<td>0.949</td>
</tr>
</tbody>
</table>
Alert Aggregation

- Girdle achieves 10.4%-72.7% improvement
  - FP-Growth [1] is vulnerable to noise and unable to address rare yet important alerts
  - UHAS [2] does not learn from history
  - LiDAR [3] uses textual similarity which is not reliable

<table>
<thead>
<tr>
<th>Method</th>
<th>Dataset1</th>
<th>Dataset2</th>
<th>Dataset3</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP-Growth</td>
<td>0.481</td>
<td>0.523</td>
<td>0.546</td>
</tr>
<tr>
<td>UHAS</td>
<td>0.697</td>
<td>0.71</td>
<td>0.707</td>
</tr>
<tr>
<td>LiDAR</td>
<td>0.742</td>
<td>0.758</td>
<td>0.826</td>
</tr>
<tr>
<td>GIRDLE</td>
<td>0.831</td>
<td>0.866</td>
<td>0.912</td>
</tr>
</tbody>
</table>

[1] Han et al. Mining frequent patterns without candidate generation. ACM SIGMOD Record ‘00.
Topic 3: Unsupervised and unified alert aggregation

- Motivation
- Graph representation learning for alert aggregation
- Evaluation
- Summary
Summary of Topic 3

- Graph representation learning for alert aggregation
  - Incomplete cascading topology of failures
  - Learn alert correlation with multi-source information
- Girdle has been deployed in production and we received positive feedback
Outline

- Topic 1: An empirical study on industrial incident management
- Topic 2: Interpretable and adaptive performance anomaly detection
- Topic 3: Unsupervised and unified alert aggregation
- Conclusion and Future work
Conclusion

Software reliability engineering
Intelligent Service Monitoring

Incident management study
- Empirical study
- Challenges and reasons of incident handling
- Thesis guidance

Log anomaly detection
- Experience report
- A toolkit for reuse
- Good performance: accurate, fast, and high-coverage

Metric anomaly detection
- Metric pattern extraction
- Interpretable results
- Adaptable to new patterns
- Interpretability and adaptivity

Alert aggregation
- Multi-source data usage
- Unsupervised alert correlation learning
- Impact scope estimation

Empirical study
Challenges and reasons of incident handling
Thesis guidance
Future Work

Current work

- Software side of the cloud, i.e., SaaS and PaaS layers

Future work

1. Network infrastructure
2. Full-stack monitoring

Service usage

Full-stack monitoring

- Network infrastructure
- Apps & Services
- VM & Containers

Software side of the cloud, i.e., SaaS and PaaS layers
Future Work (1)

Performance Monitoring and Diagnosis for Cloud Overlay Networks
  - Overlay networks are created by abstracting physical infrastructure

  - Performance monitoring via probing

  - Probing task design with the following two objectives
    - Minimum probing overhead
    - Fast diagnosis capability
Cross-layer Failure Propagation Modeling in Cloud Systems

- Existing work assumes isolated failures
  - Faults only exist in the service or layer under discussion, while others function normally
  - Not realistic in production systems

- Full-stack cloud monitoring
  - Trace problems at all cloud layers


Thank you!

Q & A