Event Correlation: A fresh approach towards reducing MTTR

Jeff Weiner
Chief Executive Officer

Renjith Rajan
Rajneesh
<table>
<thead>
<tr>
<th>Time</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>14:00</td>
<td>Introductions</td>
</tr>
<tr>
<td>14:03</td>
<td>The Problem Statement</td>
</tr>
<tr>
<td>14:08</td>
<td>Architecture Considerations</td>
</tr>
<tr>
<td>14:13</td>
<td>Event Correlation Overview</td>
</tr>
<tr>
<td>14:38</td>
<td>The Big Picture</td>
</tr>
<tr>
<td>14:43</td>
<td>Key Takeaways</td>
</tr>
<tr>
<td>14:45</td>
<td>Q&amp;A</td>
</tr>
</tbody>
</table>
Introductions

Renjith Rajan
Production-SRE
Joined LinkedIn in 2016

Rajneesh
Production-SRE
Joined LinkedIn in 2015
The Problem Statement
Problem Statement

- Reliability
- Learning Curve

Service Complexity
Problem Statement

High MTTR
Complex micro services architecture delays the identification of root cause

False Escalations
Lack of visibility into the system as whole results in false escalations
Problem Statement

Not Applicable use cases
External monitoring services show high page load times or errors

Applicable use cases
A service has high latency and error rates
Architecture Considerations
Architecture considerations

- Metrics Analysis: Real-Time, Ad-Hoc
- Alert Correlation: Correlating alerts on the Monitoring Dashboard
- Recommendation Engine: Rules Engine for collating recommendations
- Visualization: State of the system for visual correlation
- Integrations: Integration with notifications system
Event Correlation Overview
Microservices knowledge

Scattered

Not updated

Not machine friendly
How do we map service

Service Discovery
- Service
- Endpoints
- Protocol

Metrics
- Destination service
- Endpoint
- Protocol
Callgraph

- Created
  - Programmatical

- Interface
  - API and a User Interface

- Lookup
  - Service, context path, etc

- Stores
  - Callcount, latency and error rates
Site Stabilizer | Real Time and Ad-Hoc Metrics Analysis
Challenge: Not all metrics had thresholds

Challenge: expensive, real time processing, tuning based on the individual metrics behaviour

Challenge: expensive, real time processing, tuning based on the individual metrics behaviour
Clustering Algorithm

K-Means

Partitions

$n$ observations to $k$ clusters

Store

Can be trained and saved
K-Means: How it works

- Cluster center
- Cluster 1
- Cluster 2
- Cluster 3
Predict score

cluster center

cluster 3
Ranking

Predict score
- Using K-Means

Trend score
- Based on the trend of the time series

WoW
- Leverage week on week data
Typical Workflow

**Identify**

Identify the critical metrics using the k-means method

**Drilldown**

Drilldown to the corresponding critical services
Result

Top 5

Graphs
inVisualize | Alert Correlation and Visualization
inVisualize
Alert Correlation and Visualization

Polls the monitoring system continuously for alerts

Correlates downstream alerts using Callgraph

Ingests and represents callcount, average latency, error rate from callgraph
inVisualize Assumptions

Higher the alerts for a service, more likely it’s affected or broken.

Higher the callcount to a downstream, more valuable it is.

Higher the change in latency/error to a downstream, more likely it’s broken.
inVisualize
inVisualize
Alert Correlation and Visualization

- **Save** the states continuously for replay
- **Rank** the services based on a score and accessible via API
- **Score** is normalized between 0-100
Recommendation Engine
**Recommendation Engine**

**Input**
- Service, colo, duration

**Collate**
- Collates the outputs from Site stabilizer and inVisualize

**User Interface**
- Responsible service, SRE team, correlation confidence score

**Decorate**
- With information such as scheduled changes, deployments and A/B experiments
Event Correlation | The Big Picture
The Big Picture

Latency Alert

Nurse Plan arguments
• service-name: my-frontend
• req_confidence = 85
• escalate = True

Escalate to correct SRE

Find what’s wrong with ‘my-frontend’ in DatacenterB

Service: Service-C
Confidence: 91%
Reason: ‘Service-C’ has high latency after a deploy
Service Owner: SRE

Event Correlation API

Iris

Escalate to correct SRE
Key Takeaways
Key Takeaways

- **Learning Curve**
  - Microservices knowledge

- **SRE Workload**
  - Reduced MTTR helps bring down the SRE workload

- **Escalations**
  - Reduced On-Call Escalations

- **Flexibility**
  - Comparison data set creation plans

- **Thresholds**
  - Dynamic