September 2023

Aggregated End-to-end Visibility and its Application on Rapid and Automatic Outage Triage in Monitoring Microservices

Kyusoon Lee

Thomas Adamcik

Parker Roth

Jan-Jan van der Vyver XIV

Mayur Kabra

Aggregated End-to-end Visibility and its Application on Rapid and Automatic Outage Triage in Monitoring Microservices

ABSTRACT

In a microservice architecture, a user request can go through a large number of servers owned by several different teams before a response is returned. The request can fail due to failure in any of the servers. Troubleshooting an outage that affects the end user experience in microservice architecture can involve multiple teams and can take a substantial amount of time. This disclosure describes techniques to rapidly locate the root cause entity of a customer-facing failure to node(s) deep within the infrastructure of the service. Per the techniques, end user product teams mark requests with metadata known as critical user interactions (CUI). The metadata is propagated along with the request. Performance metrics are gathered from servers that the requests go through. The performance metric is keyed by CUI, server node, and peer node for every adjacent pair of nodes. These piecemeal metrics keyed by CUI together offer end-to-end visibility for a set of requests grouped by the CUI of the end product, enabling the rapid and automatic triage of an outage to an interior server without requiring domain expertise on the product or the server.

KEYWORDS

- Microservice
- Telemetry
- Distributed tracing
- Dependency analysis
- Reliability engineering
- Root cause analysis
- Queries per second (QPS)

BACKGROUND

In a microservice architecture, a user request can go through a large number of servers owned by several different teams before a response is returned. The request can fail due to failure in any of the servers. Infrastructure teams find it difficult to map outages that they observe to end user impact. End user product teams cannot point to a backend node as a root cause of an outage easily. Troubleshooting an outage that affects the end user experience in microservice architecture can involve multiple teams and can take a substantial amount of time.

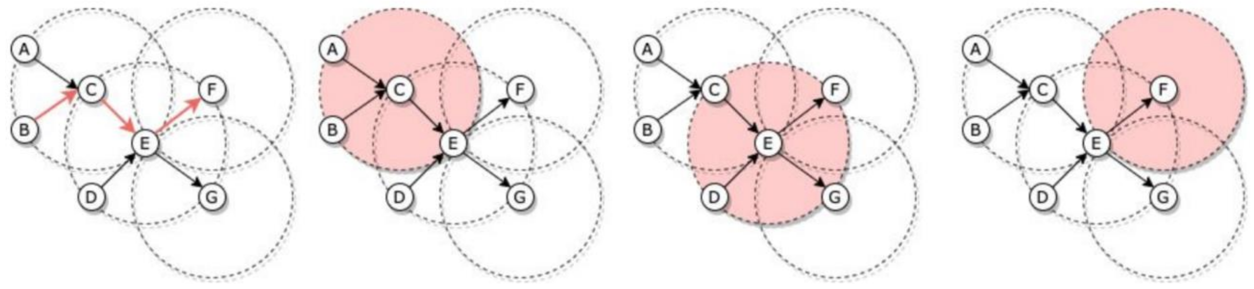

Fig. 1: As a request flows through a network, its origins become obscure

Fig. 1 illustrates how traditional server-centric observability obscures the origins of a request as the request flows through a network. The request originates at node (or server) B, follows the path B→C→E→F, and terminates at node (server) F. As far as node F is concerned, the request originated from node E. Upstream of node E, there is uncertainty as to whether the request originated from node C or node D. Similarly, upstream of node C, there is uncertainty as to whether the request originated from node A or node B. Thus, as far as the terminal node F is concerned, the request may have originated at any one of nodes A, B, C, E, or D. Similarly, as far as the origin node B is concerned, the error that it observes may have originated at any one of nodes C, E, F, or G. As the network depth increases, the number of involved nodes increases exponentially, and the uncertainty concerning the origin of a request or an error grows.

Typical root cause analyses start from the end user product, where customer-facing symptoms are observed. Yet, as discussed above, due to the uncertainty concerning the origin of error beyond the nearby nodes, determining the root cause entity where the customer-facing symptoms started from is at best inefficient.

Traditional server metrics offer insight into all the requests visible to a given server. However, the view is limited to the server. Backends far away from end user products cannot tell which end user product or what kind of critical user interaction is impaired. End user product teams cannot determine the particular backend among the large number of potential backends that caused the failure.

Distributed tracing, a technique that propagates a unique, per-request ID along with the request, enables greater visibility for a request. However, this view is limited to a single request. The outage being investigated may not be affected by the request. The troubleshooting team has to take a chance on picking a relevant and representative distributed trace for the outage being investigated.

DESCRIPTION

This disclosure describes techniques to create an aggregated end-to-end view of a request traveling through a microservice architecture. The view enables triaging an outage to a server without requiring domain expertise on the server. A root cause entity of a customer-facing outage can be rapidly and automatically tracked down to a malfunctioning node that is deep within the infrastructure. The duration and frequency of outages can be reduced.

Aggregated End-to-end Visibility

Per the techniques, end user product teams mark user requests with metadata referred to as critical user interactions (CUI). An example format of the CUI can be

<product>/<interaction>. For example, if a request is to watch a video on a web service hosted at a particular domain e.g., sampledomain.com, then the metadata can be a string such as sampledomain/WATCH. The metadata is propagated along with the request through the stack using standard distributed tracing protocols.

Performance metrics such as latency or error ratio are gathered from servers that the CUI and the request go through. The performance metric is keyed, or broken down, by CUI for every pair of the two adjacent nodes. Such metrics together offer aggregate end-to-end visibility for a set of requests grouped by the CUI of the end product.

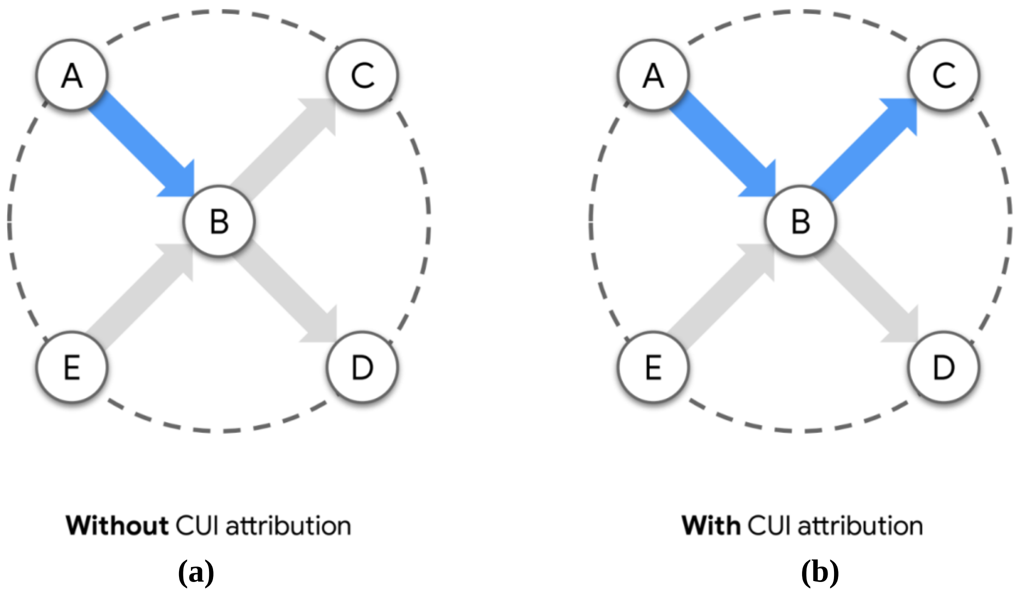

Fig. 2: Root-cause analysis is accelerated with CUI attribution

Fig. 2 illustrates how aggregate end-to-end visibility enables a systematic approach to identify root-cause entities multiple hops away. Without such visibility (Fig. 2a), node A (the caller node of node B) sees node B returning an error. Node B believes the errors are coming from its backends, but it has no means to tell whether node A's errors in particular are coming from node C or node D, because for example, it is also serving E's requests. By contrast, with aggregated end-to-end visibility (Fig. 2b), the exact error ratio for the CUI between every pair of adjacent nodes is known. Hence, it is feasible to determine the path, A→B→C that the anomaly (e.g., errors) is returned from, and the root-cause entity (node C) that the anomaly started from.

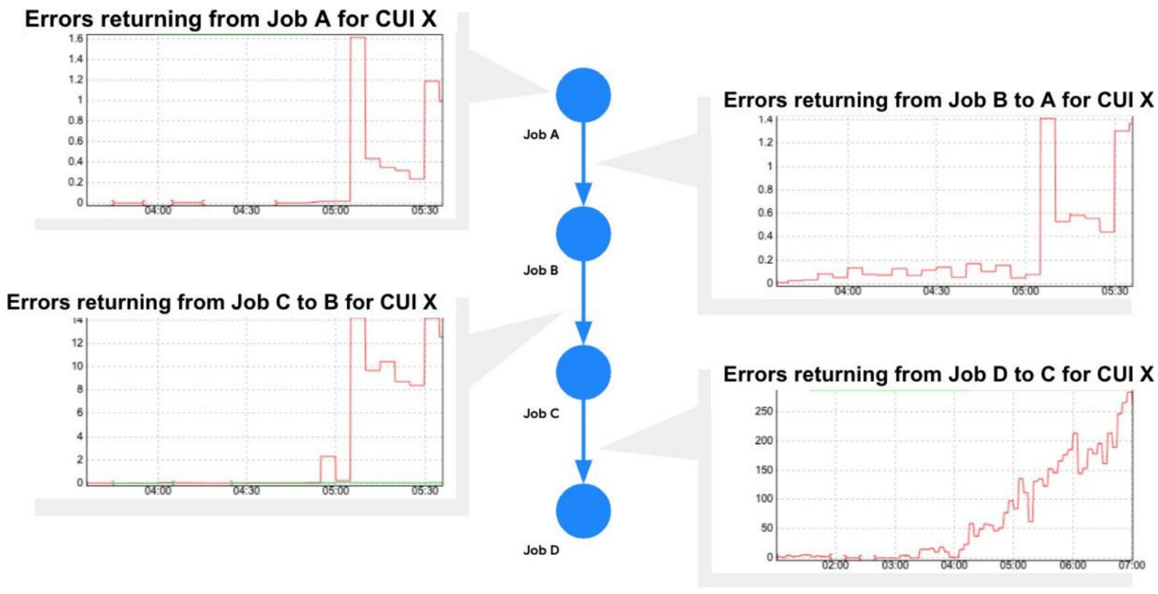

Fig. 3: Error rates increase with reducing distance from the root-cause node

As illustrated in the example of Fig. 3, node (or job) A, which can be an end-user product, sees an increased error rate against time. Furthermore,

- node A sees an increased error rate being returned from node B (B→A error rate) for a particular CUI X;
- node B sees an increased error rate being returned from node C (C→B error rate) for the CUI X;
- node C sees an increased error rate being returned from node D (D→C error rate) for the CUI X.

The increase in error rate along the path A→B→C→D indicates that the root cause entity of the problem seen at node A is likely located at node D.

It is possible that jobs running on the different nodes require markedly different domain expertise to debug. Furthermore, the different nodes may belong to different teams. Even so, per the techniques described herein, root-cause entity identification can be made without custom configuration or consulting domain experts of each node along the pathway.

Automatic triaging of an outage to a root cause

As shown above, aggregated end-to-end visibility from CUI tagging enables viewing of otherwise concealed anomalies and linking them together. To automatically triage an outage to a root cause entity, a formal definition of anomaly, also referred to as outage, follows.

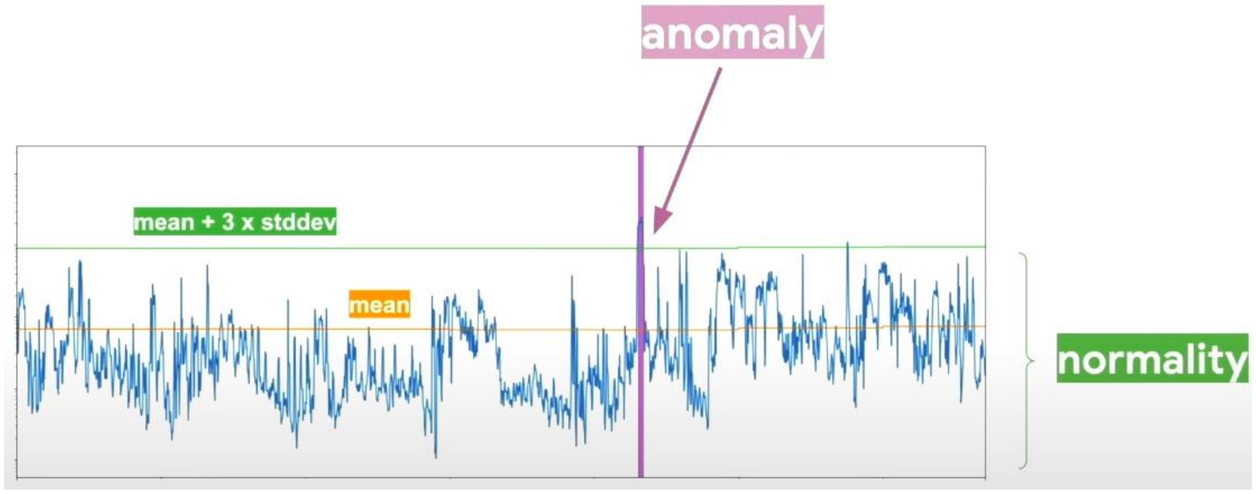

Fig. 4: Anomaly can be defined as sustained divergence from a normal range of operation

A range of normal operation between two adjacent nodes is defined by the mean and standard deviation of the number of errors (or other performance metric such as latency or error ratio) between the nodes for a given CUI for a preceding period, e.g., the last several weeks, the last thirty days, etc. The mean and standard deviation can be updated periodically, e.g., daily. An anomaly is defined as sustained divergence from the normal range. For example, as illustrated in Fig. 4, an error ratio that falls within mean plus three times standard deviation can be considered as normal. A median error ratio over a certain time window (e.g., thirty minutes) that exceeds the mean plus three times standard deviation can be considered as an anomaly.

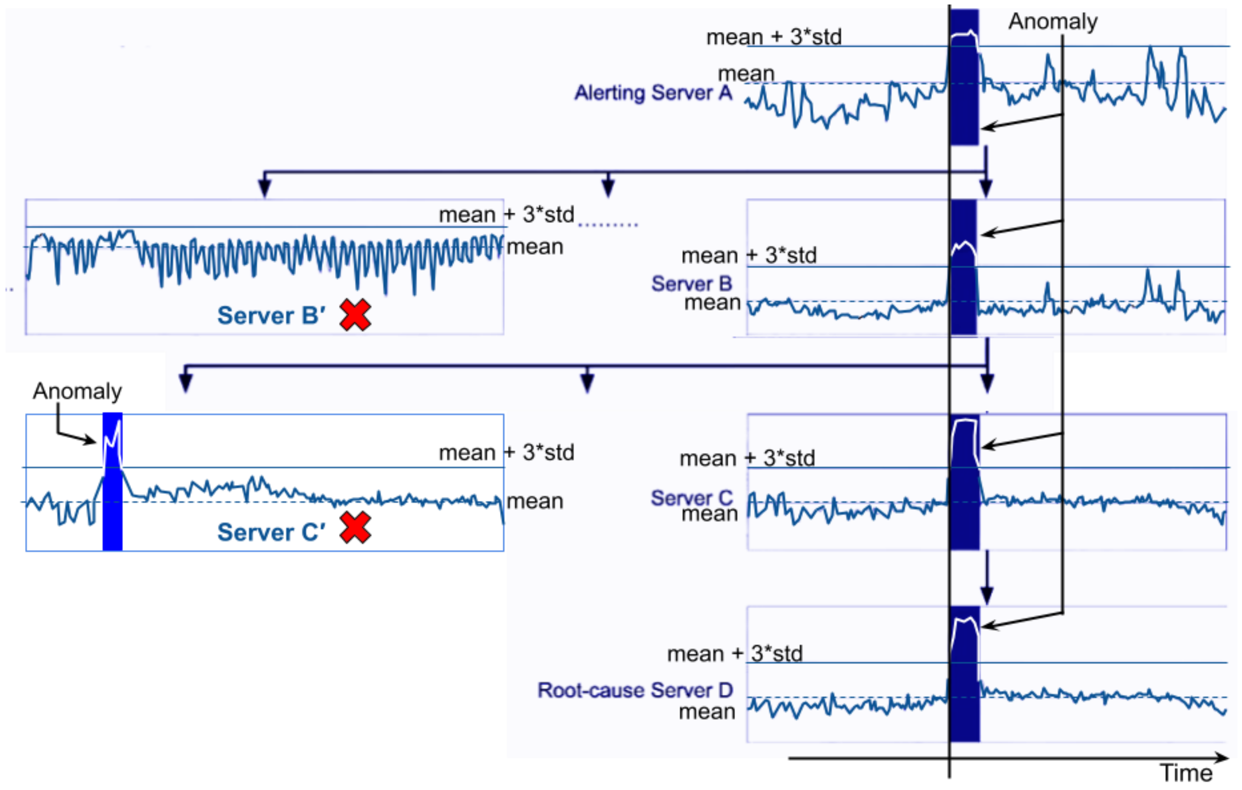

Fig. 5: Automatic triaging of an outage to a root-cause server

Fig. 5 illustrates automatic triaging of an outage to a root-cause entity. Server A, which can be a customer-facing server (referred as “alerting server”), experiences an outage, e.g., a sustained burst of errors that is more than three standard deviations from the mean error. The dependencies (or downstream nodes on the CUI path) of server A are server B and server B′. Server B′ is found to experience no anomaly and is ruled out from the chain of anomalies. Server B is found to experience an anomaly at nearly the same time as server A and is included in the chain of anomalies.

Among the dependencies of server B, server C′ and server C are found to experience anomalies. However, server C′ experiences the anomaly at a different time than the anomaly at server B and is hence ruled out. Server C experiences anomaly at nearly the same time as server B and is hence included in the chain of anomalies.

Among the dependencies of server C, server D is found to experience an anomaly at nearly the same time as server C. Server D either has no dependencies, or, if it does, its dependencies show no anomalies. Server D is thus identified as the root cause of the issues being experienced at the customer-facing server A. In this manner, an anomaly chain A→B→C→D is established between the customer-facing server and the root-cause entity.

The identity of the root-cause entity along with the anomaly chain is automatically sent to the relevant teams for damage control and recovery. The teams can leverage a shared understanding of the outage to execute a rapid, coordinated recovery. Alerts can be meaningfully targeted. Alerts are not sent to teams that are not along the anomaly chain, thereby saving those teams from wasted effort. Alerts can be also sent with different urgency to different teams depending on the involvement of the teams. It is possible that a given customer-facing outage has more than one root cause; if so, the described procedure determines the multiple root causes. The traditional procedure, which is for human experts to manually reconstruct the anomaly chain (or incident tapestry) using fragmented information collected across the network, is obviated.

The procedure illustrated graphically in Fig. 5 can be described formally as follows.

- Periodically (e.g., daily) a range of normal operation between two adjacent nodes is updated.
- Run the above-described anomaly detection algorithm for every pair of source node and destination node used for every CUI in real-time.

- Choose a combination of CUI and compute-cluster job at an entry point.
- Fetch all the pairs of adjacent nodes with anomaly detected for the CUI.
- Traverse the tree starting from the entry point.
- The leaf nodes where the traversal ends are the root-cause entities.

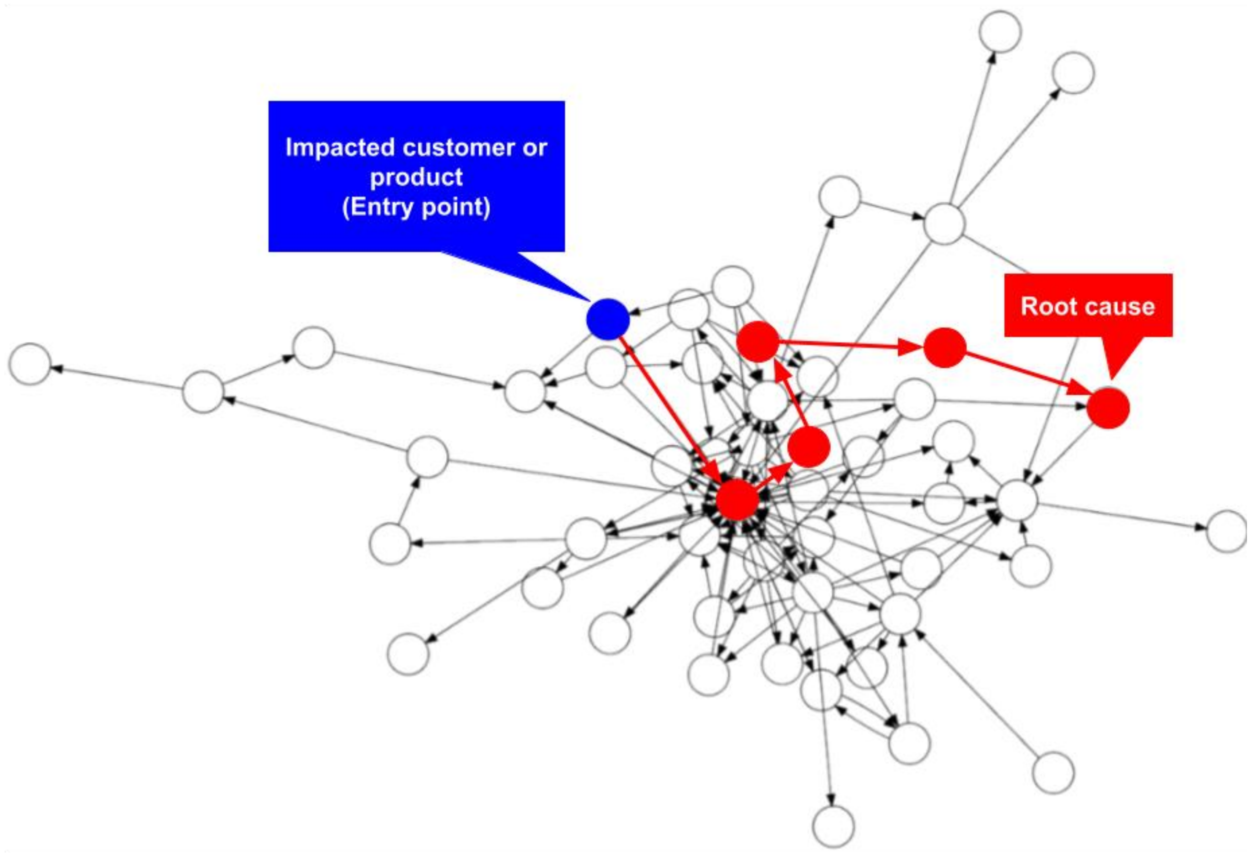

Fig. 6: Overlaying the anomaly chain over network topology

As illustrated in Fig. 6, root-cause analysis and determination can be visualized in real time in graphical form by overlaying the anomaly chain (or the chain of node performance metrics) over the network topology. The example of Fig. 6 shows the anomaly chain traced by a single CUI. Other CUIs can have their own anomaly chains overlaid over the same topology. A chain of anomalies can end in multiple root causes.

CONCLUSION

This disclosure describes techniques to rapidly locate the root cause entity of a customer-facing failure to node(s) deep within the infrastructure of the service. Per the techniques, end user product teams mark requests with metadata known as critical user interactions (CUI). The metadata is propagated along with the request. The performance metric is keyed by CUI, server node, and peer node for every adjacent pair of nodes. These piecemeal metrics keyed by CUI together offer end-to-end visibility for a set of requests grouped by the CUI of the end product, enabling the triaging of an outage to an interior server without requiring domain expertise on the product or the server.

REFERENCES

1. "Propagation format for distributed context: Baggage," available online at https://www.w3.org/TR/baggage/ accessed September 9, 2023.
2. "Beyond Distributed Tracing | USENIX," available online at https://www.usenix.org/conference/srecon22americas/presentation/lee accessed September 13, 2023.
3. "The future of Google Monitoring, powered by CUI attribution | Container Solutions", available online at https://www.youtube.com/watch?v=qFTVJtlj0xo accessed on September 22, 2023.