AlwaysOn Profiling

with

Splunk Application Performance Monitoring (APM)

splunk> turn data into doing™

Agenda - APM AlwaysOn Profiling

1. Use Case Review
2. Documentation and OTEL Configurations
3. AlwaysOn Profiling Storyboard Overview
4. AlwaysOn Profiling Scenario Solution
5. In Summary, what did we learn?

Situation Statement:

CSCorp (Our fictitious company) wants to improve the overall quality of services delivered by optimizing service code performance and resource consumption. They want to be more proactive and introduce process and governance into their DevOps SDL (Software Development Lifecycle) to support this initiative.

<u>This case looks at how we can use APM AlwaysOn Profiling workflow initiated from a service or application performance detector alert.</u>

As part of this new process strategy, you might have some questions as these:

- How can an SRE proactively use Splunk AlwaysOn Profiling insights to spot potential code performance bottlenecks?
- How can a Developer use Splunk AlwaysOn Profiling tooling to isolate bottlenecks in code that may be contributing to high CPU utilization and/or memory consumption?

Scenario

- The OnlineBoutique application consumes many microservices to run the online sales portal. CSCorp's DevOps team has introduced a process where SRE's can identify opportunities for code optimization. A critical service called 'adservice' has experienced some performance degradations that are adversely impacting customer experience on the Online Boutique portal. They also are seeing some trending performance degradations over time that are not impacting customers currently and are not breaching SLA, but proactively the SRE believes a code optimization may be beneficial.

Goals:

- For an SRE to easily identify code based performance bottleneck suspects and submit a code optimization ticket with an attached flame graph call stack export to be used by developer to hone in on the potential bottleneck in code.
- For a developer to effectively and efficiently use outputs from flame graph analysis to navigate and identify code inefficiencies, leading to code optimization, improved performance and developer productivity.

Personas:

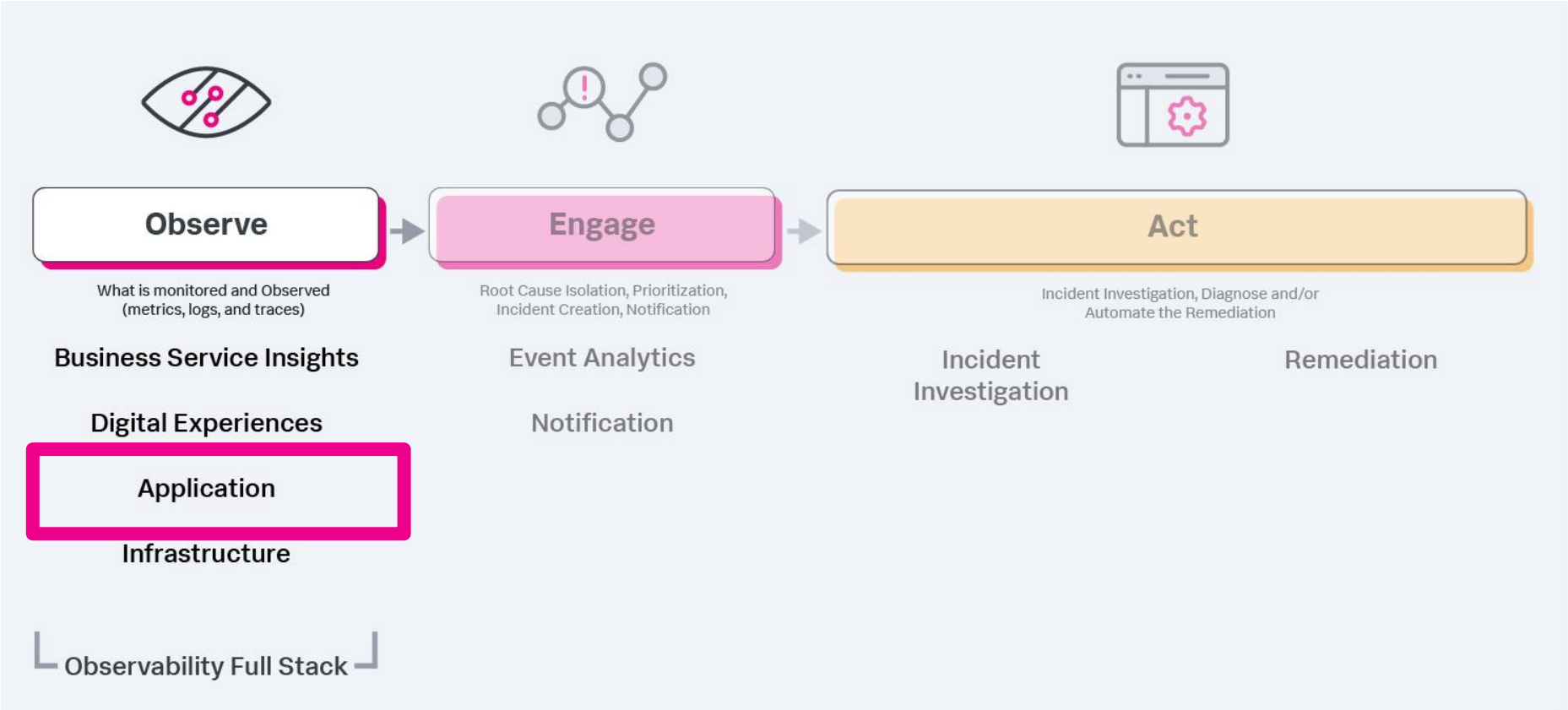
- Persona 1: SRE responsible for OnlineBoutique's customer digital experience and the application performance and availability
- Persona 2: Service Developer - Building and deploying microservice code releases. In this article we will be using 'adservice' of OnlineBoutique app.

Assumptions/statements:

- ‘adservice’ microservices are critical services at CSCorp
- ‘Frontend’ service is the main entrypoint gateway for OnlineBoutique app.

Value

- SRE - Service Quality improvements via code optimization - provides process and tooling to identify potential opportunities for code optimization leading to improvements in service performance and resource usage efficiencies,
- Developers - productivity improvements - accelerate isolation of code where performance bottlenecks are occurring.
- Reduction in required capacity per workload, resulting in overall compute bill reductions.



Documentation References

APM Product Reference

https://docs.splunk.com/Observability/profiling/intro-profiling.html#nav-Introduction-to-AlwaysOn-Profiling

Get AlwaysOn Profiling data into Splunk APM

Prerequisites

To get data into Splunk AlwaysOn Profiling, you need the following:

- Splunk APM enabled for your Observability Cloud organization.
- The Splunk Distribution of OpenTelemetry Collector version 0.44.0 or higher. See Install and configure the Splunk Distribution of OpenTelemetry Collector.

If the version of your Splunk OTel Collector is lower than 0.44.0, see Check the OpenTelemetry Collector configuration.

Get profiling data in

Follow these instructions to get profiling data into Splunk APM using AlwaysOn Profiling:

1. Instrument your application or service.
2. Enable AlwaysOn Profiling.
3. Check that Observability Cloud is receiving profiling data.

Instrument your application or service

AlwaysOn Profiling requires APM tracing data to correlate stack traces to your application requests. To instrument your application for Splunk APM, follow the steps for the appropriate programming language:

Language	Available instrumentation	Documentation
Java	Splunk Distribution of OpenTelemetry Java v1.12.0 or higher	Instrument a Java application for Splunk Observability Cloud
Node.js	Splunk Distribution of OpenTelemetry JS	Instrument a Node application for Splunk Observability Cloud
.NET	SignalFx Instrumentation for .NET	Instrument a .NET application for Splunk

Splunk-otel-collector (Agent)

- Need to run otel collector version **0.44+ (August 2022, latest v0.58.0)**
- If installing via helm charts, make sure to pass as shown below
  `--set="splunkObservability.profilingEnabled=true"`

```
helm install splunk-otel-collector \
--set="splunkObservability.realm=$REALM" \
--set="splunkObservability.accessToken=$ACCESS_TOKEN" \
--set="clusterName=$(hostname)-k3s-cluster" \
--set="splunkObservability.logsEnabled=true" \
--set="splunkObservability.profilingEnabled=true" \
--set="environment=$(hostname)-apm-env" \
splunk-otel-collector-chart/splunk-otel-collector \
-f ~/workshop/k3s/otel-collector.yaml
```

Splunk-otel-java (Auto-Instrumentation)

- Needs **JDK 11+** or **JDK 8u262+**
- Agent version **1.14.2+** required
- To enable profiling:
  - `-Dsplunk.profiler.enabled=true`
  - `-Dsplunk.profiler.memory.enabled=true`
  - `-Dotel.exporter.otlp.endpoint=http(s)://collector:4317`
  - `-Dsplunk.metrics.endpoint=http(s)://collector:9943`
- Full configuration reference: https://docs.splunk.com/Observability/gdi/get-data-in/application/java/configuration/advanced-java-otel-configuration.html#profiling-configuration-java

Java instrumentation - Helm Chart YAML

Sample K8S deployment snippet

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: adservice
spec:
  selector:
    matchLabels:
      app: adservice
  template:
    metadata:
      labels:
        app: adservice
spec:
 terminationGracePeriodSeconds: 5
      tolerations:
      nodeSelector:
      containers:
        - name: server
          image: quay.io/phagen/adserviceprofiling:4.0
          imagePullPolicy: Always
```

```yaml
        ports:
          - containerPort: 9555
         env:
          - name: PORT
            value: '9555'
          - name: OTEL_SERVICE_NAME
            value: adservice
          - name: OTEL_RESOURCE_ATTRIBUTES
            value: "deployment.environment=vzmp-apm-env"
          - name:  OTEL_PROPAGATORS
            value: "b3multi"
          - name: NODE_IP
            valueFrom:
              fieldRef:
                fieldPath: status.hostIP
          - name: SPLUNK_PROFILER_ENABLED
            value: "true"
          - name: SPLUNK_PROFILER_MEMORY_ENABLED
            value: "true"
          - name: SPLUNK_METRICS_ENABLED
            value: "true"
```

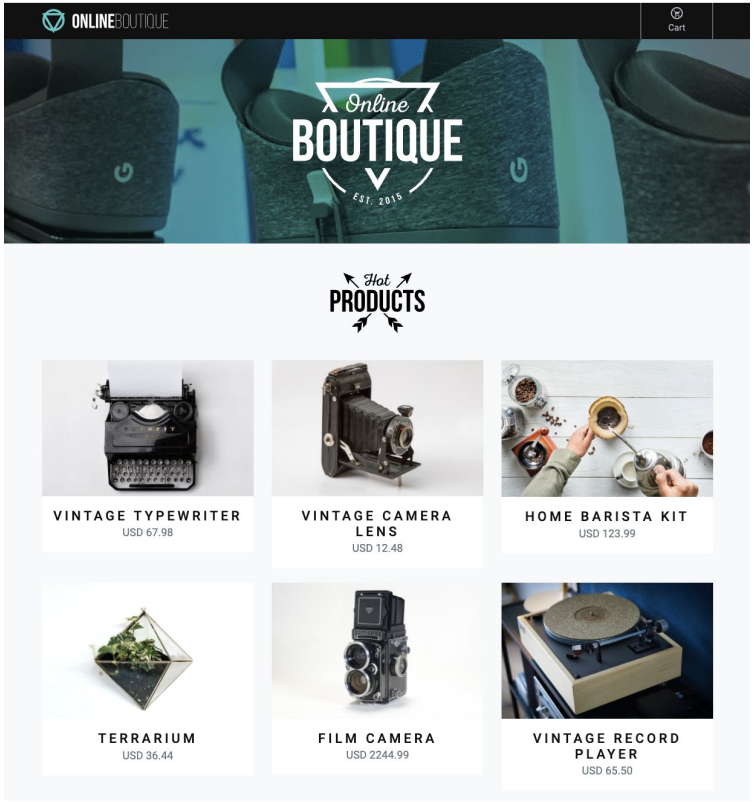
Splunk-otel-js [node.js] (Auto-Instrumentation)

- Needs **Node 12.3+**
- Agent version **1.2.1+** required
- To enable profiling:
  - `SPLUNK_PROFILER_ENABLED=true`
  - `SPLUNK_PROFILER_LOGS_ENDPOINT=http(s)://collector:4317`
- Full configuration reference: https://docs.splunk.com/Observability/gdi/get-data-in/application/nodejs/configuration/advanced-nodejs-otel-configuration.html#profiling-configuration-nodejs

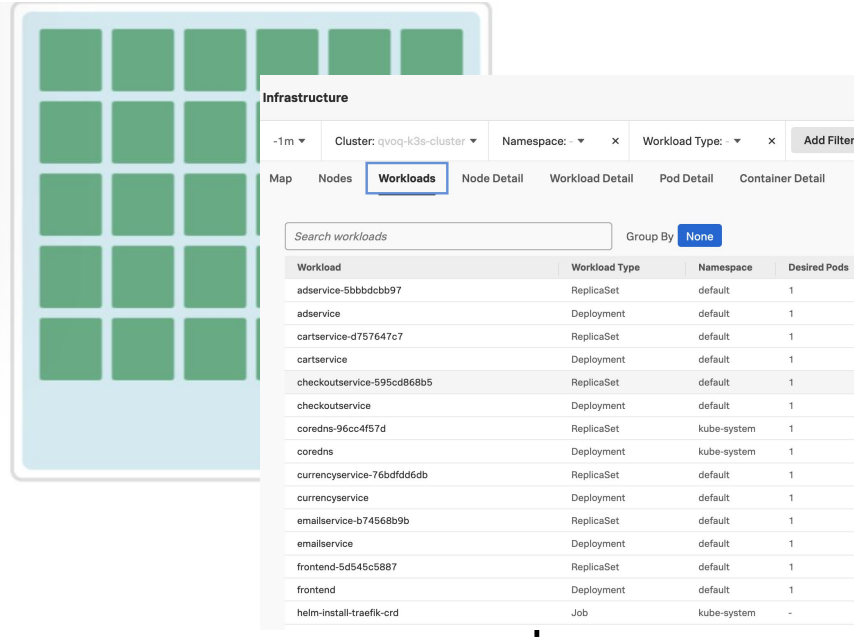
Signalfx-dotnet-tracing [.net] (auto-instrumentation)

- Needs **.NET Core 3.1+** or **.NET 5.0+**
- Agent version **0.2.7+** required
- To enable profiling:
  - `SIGNALFX_PROFILER_ENABLED=true`
  - `SIGNALFX_PROFILER_LOGS_ENDPOINT=http(s)://collector:4317`
- Full configuration reference: https://docs.splunk.com/Observability/gdi/get-data-in/application/dotnet/configuration/advanced-dotnet-configuration.html#profiling-configuration-dotnet

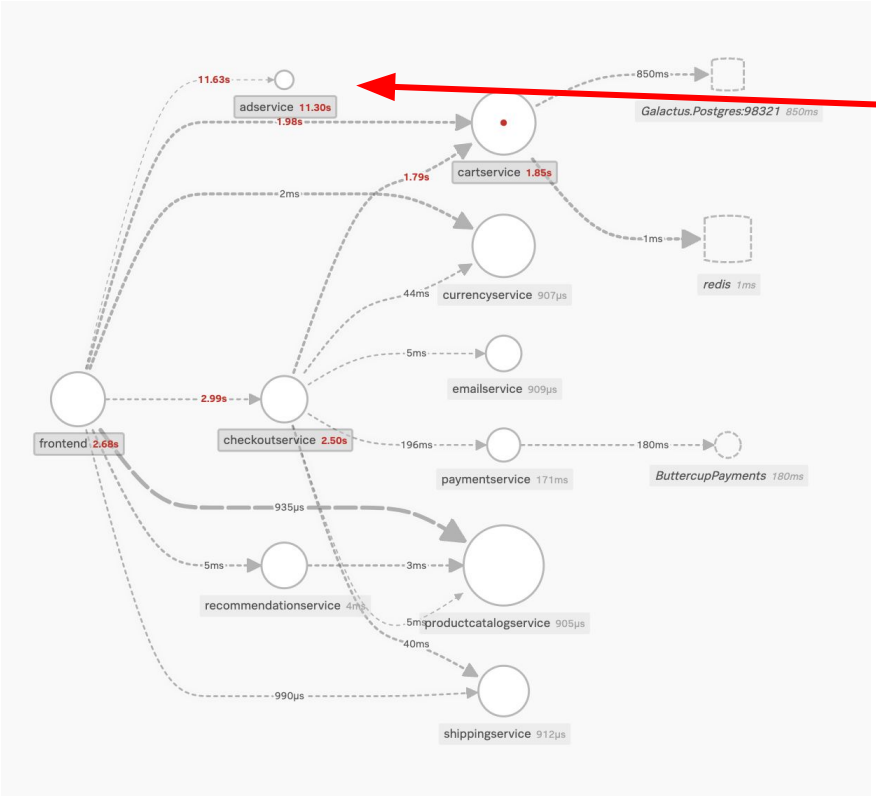
Online Boutique Application



Hosted on Kubernetes Cluster Monitored by Splunk IM

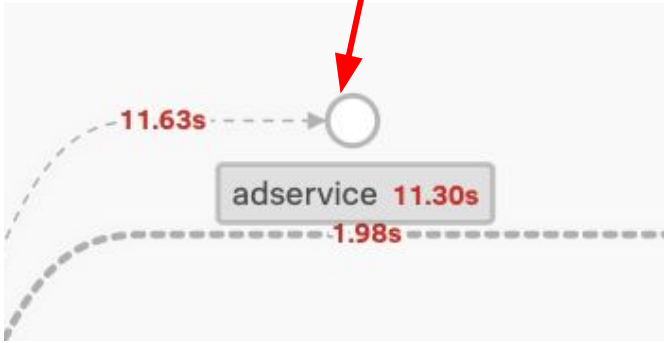


Online Boutique Monitored by Splunk APM



Splunk APM Service Map

Focus on ‘adservice’ for AlwaysOn Profiling



AlwaysOn Profiling Storyboard Overview

SRE Persona

1. APM Detector fires on latency breach and notifies the Online Boutique SRE team.
2. SRE Investigates the alert and starts the triage process to determine probable cause.
3. SRE Determines there are traces that are taking a long period of time and determines there may be a code bottleneck in java code based upon the flame graph visualization.
4. SRE takes a copy of the stack trace, opens a Jira ticket and attaches the stack trace snippet for the Developer of the service to engage.

Shifts to Developer Person

AlwaysOn Profiling Storyboard Overview

Developer Persona

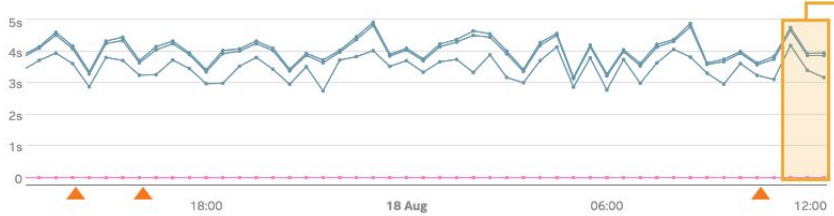
5. Developer assigns themselves to ticket and goes directly into the Alert in the context of the service (adservice) and time.
6. Developer explorers traces, stack traces and alwaysOn proofing UI for CPU and Memory.
7. Developer within their Java IDE, imports the stack trace file from the ticket
8. Develop uses this to isolate the function/method that appears to be causing excessive latency causing CPU spikes in the K8s container hosting the workload.
9. Developer makes the changes and pushes changes into the CI/CD pipeline for test and release of ‘adservice’ and monitors for improvements

APM Type detector on ‘adservice’ - latency

Edit Alert Rule - Static Threshold - [APM] Service Latency (median) -... 30m

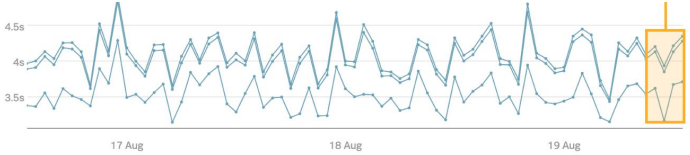
Time -1d

Estimated alert count: 3 in 1 day. Alerts that would have triggered ▲ shown in chart below.



✓ Type	APM Rule
✓ Alert signal	[APM] Service Latency (median) - [adservice:*], sf_kind:SERVER,CONSUMER, sf_service:adservice, sf_operation:*, sf_environment:vz…
✓ Alert condition	Static Threshold
✓ Alert settings	The 90th percentile of latency is above 3000ms for 100% of 10s. Clears when below 1000ms for 100% of 1m.
⚠ Alert message	Severity: Major Runbook URL: None Tip: None
✓ Alert recipients	No recipients
7. Activate...	Rule Name Request Latency distribution Detector Update Alert Rule

APM Type detector on ‘adservice’ - latency Historical Anomaly detection



- ✓ Type
- ✓ Alert signal
- ✓ Alert condition
- ✓ Alert settings
- ⚠ Alert message
- ✓ Alert recipients
- 7. Activate...

Customize the settings for this alert

Alert trigger: The 90th percentile of latency in the last 15m is more than 20% above its historical norm, cyclical over 1w periods. Clears when latency is less than 5% above norm. Learn more

Alert Settings | Simulated Events | Data Table

Cycle length: 1w

Trigger sensitivity: ○ Low ○ Medium ○ High ◉ Custom

Auto-Clear alerts: ☑ Clear active alerts if metric time series has not reported for 1h

Hide advanced settings

Signal resolution: 10s

Alert based on: Percentage change

Percentile to monitor: 90

Current window: 15m

Historical window: 1h

Number of previous cycles: 4

Trigger threshold (%): 20 %

Clear threshold: 5 %

Exclude errors?: Yes

Min. req/sec (absolute): 5

Min. req/sec (% of history): 50 %

Customize the settings for this alert

Alert trigger: The 90th percentile of latency in the last 15m is more than 20% above its historical norm, cyclical over 1w periods. Clears when latency is less than 5% above norm. Learn more

Alert Settings | Simulated Events | Data Table

Pinned Valu...	Value	Rollup	Plot Name	sf_metric	sf_kind	sf_environ...	sf_operatio...
1,450,000	1,450,000		BT	spans.duration.ns.p90		vzmp-apm-...	hipstershop....
1,450,000	1,450,000		BU	spans.duration.ns.median		vzmp-apm-...	hipstershop....
1,450,000	1,450,000		BS	spans.duration.ns.p99		vzmp-apm-...	hipstershop....

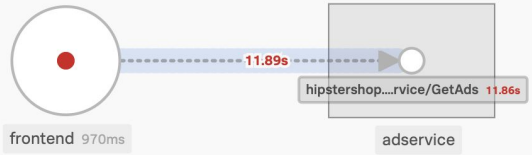
SRE Investigates Alert from Detector

1

0	1	0	0	0
Critical	1 Major	Minor	Warning	Info

Rule Name and Source	Detector Name	Duration
▲ Request Latency distribution Detector - hipstershop.AdService/GetAds vzmp-apm-env SERV…	OB:adservice:Request Latency Dete…	30 minutes

3



RUM Hipstershop Ad Service Get Ads Error Rate (value: 13,028) > 3000

Plots: d: metric_name (value: 13,028) Trigger Threshold (value: 3,000)

Show plot information

Message

Rule "Request Latency distribution Detector" in detector "OB:adservice:Request Latency Detector" triggered at Fri, 19 Aug 2022 17:43:40 GMT.

Triggering condition: The 90th percentile of latency is above 3000ms for 100% of 10s. Clears when below 1000ms for 100% of 1m.

Latency: 13028.421999999999ms Threshold: 3000ms Signal details: {sf_environment=vzmp-apm-env, sf_kind=SERVER, sf_operation=hipstershop.AdService/GetAds, sf_service=adservice}

About this Alert

Incident ID FahY8WaA4AI

Next Steps

Tip *No tip has been set for this detector*

Runbook *No runbook has been set for this detector*

Explore Further

APM Troubleshoot

2

4

Traces

5

AlwaysOn Profiling

Top Frames

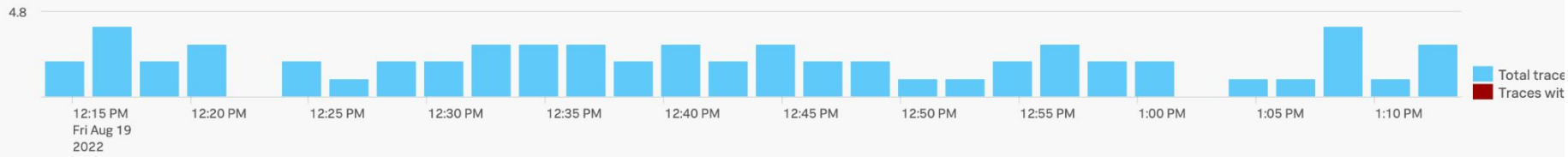
	Count
sun.nio.ch.EPollArrayWrapper.epollWait(Native Method:-1)	60.9k
java.lang.Object.wait(Native Method:-1)	21.2k
java.net.SocketInputStream.socketRead0(Native Method:-1)	13.9k
hipstershop.copyright.StockPhotos$CopyrightPhoto.<init…	1.3k
hipstershop.copyright.StockPhotos.access$100(StockPh…	97

SRE Investigates Traces

Today, 12:13pm - 1:13pm | vzmp-apm-env (1) | All Workflows | Services · 1 | Tags

Traces | Duration (ms) 3000 | Max | Errors Only | View Trace ID

62 matched traces



Trace ID	Timestamp	Duration	Initiating Operation	Services
27ed73bda3b278a8	Fri, Aug 19 2022 1:12:57 PM CDT	13.03s	frontend: GET /product/{id}	frontend (12) productcatalogservice (7) recommendationservice (2) cartservice (2) adservice currencyservice redis
269d3b4117941388	Fri, Aug 19 2022 1:12:40 PM CDT	13.05s	frontend: GET /product/{id}	frontend (12) productcatalogservice (7) recommendationservice (2) cartservice (2) adservice redis

SRE Investigates Trace and Spans

1

Span Operation

Waterfall | Span Performance

Search Spans

Matches only

28 / 28 Spans Loaded

0ms | 3.48s | 6.96s | 10.45s | 13.93s

productcatalogservice: /hipstershop.ProductCatalogService/GetProduct — 476µs

frontend: grpc.client — 13.10s

adservice: hipstershop.AdService/GetAds — 13.10s

2

Span time

Dots indicate the call stacks taken

Span Details | AlwaysOn Profiling

AlwaysOn Profiling Stack Traces 8/5067

Previous | Next

Click through stack traces

Service	adservice View in AlwaysOn Profiler
Thread State	RUNNABLE
Thread Name	grpc-default-executor-214
Snapshot Timestamp	2022-09-06T14:51:45.029000

4

Memory Allocation Stacktrace

Copy to Clipboard

```
hipstershop.copyright.StockPhotos$CopyrightPhoto.<init>(unknown:54)
hipstershop.copyright.StockPhotos.createDatabase(unknown:37)
hipstershop.copyright.StockPhotos.isCopyright(unknown:22)
hipstershop.copyright.PhotographyCertifier.certify(unknown:16)
hipstershop.copyright.CopyrightCertification.certify(unknown:32)
hipstershop.AdService.getAdsByCategory(unknown:140)
hipstershop.AdService.access$300(unknown:41)
hipstershop.AdService$AdServiceImpl.getAds(unknown:114)
hipstershop.AdServiceGrpc$MethodHandlers.invoke(unknown:205)
io.grpc.stub.ServerCalls$UnaryServerCallHandler$UnaryServerCallListener.onHalfClose(unknown:182)
io.grpc.PartialForwardingServerCallListener.onHalfClose(unknown:35)
io.grpc.ForwardingServerCallListener.onHalfClose(unknown:23)
io.grpc.ForwardingServerCallListener$SimpleForwardingServerCallListener.onHalfClose(unknown:40)
io.grpc.Contexts$ContextualizedServerCallListener.onHalfClose(unknown:86)
io.opentelemetry.javaagent.shaded.instrumentation.grpc.v1_6.TracingServerInterceptor$TracingServerCall$TracingServerCallListener.onHalfClose(unknown:133)
io.grpc.internal.ServerCallImpl$ServerStreamListenerImpl.halfClosed(unknown:340)
io.grpc.internal.ServerImpl$JumpToApplicationThreadServerStreamListener$1HalfClosed.runInContext(unknown:866)
io.grpc.internal.ContextRunnable.run(unknown:37)
io.grpc.internal.SerializingExecutor.run(unknown:133)
java.util.concurrent.ThreadPoolExecutor.runWorker(unknown:1149)
java.util.concurrent.ThreadPoolExecutor$Worker.run(unknown:624)
java.lang.Thread.run(unknown:750)
```

Top of stack : What is happening now

3

Intermediate Stack Fromes: Leading up to now

Bottom of stack : How it started

SRE views the AlwaysOn Profiler - CPU

Today, 11:46am - 12:01pm | All Environments | 1 | ervice | All Thread States | Host | Container | Process | Linked to spans | 2 | Clear A

AlwaysOn Profiling | adservice (Java) | CPU | Memory

Search stack frames

Using 558 call stacks out of 558 total

3

Name	Self time	Count
hipstershop.copyright.StockPhotos$CopyrightPhoto.<init>(StockPhotos.java:54)	7.6min	469
hipstershop.copyright.StockPhotos$CopyrightPhoto.matchesAd(StockPhotos.java:63)	55.00s	55
java.util.LinkedList.linkLast(LinkedList.java:142)	13.00s	13
java.util.Random.seedUniquifier(Random.java:116)	10.00s	10
hipstershop.copyright.StockPhotos.createDatabase(StockPhotos.java:37)	9.00s	491
hipstershop.copyright.StockPhotos.access$100(StockPhotos.java:13)	5.00s	495
java.util.Random.<init>(Random.java:137)	4.00s	4
sun.nio.ch.EPollArrayWrapper.epollWait(Native Method:-1)	1.00s	1
io.netty.util.concurrent.DefaultPromise.checkNotifyWaiters(DefaultPromise.java:628)	3ms	3
sun.nio.ch.FileDispatcherImpl.writev0(Native Method)	1ms	1
io.netty.handler.codec.http2.HpackEncoder.add(HpackEncoder.java:424)	1ms	1
io.grpc.internal.AbstractStream$TransportState.onSendingBytes(AbstractStream.java:306)	1ms	1
java.lang.Thread.run(Thread.java:750)		552
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)		551

CPU Stack Traces

Thread.run(Thread.java:750)
ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
SerializingExecutor.run(SerializingExecutor.java:133)
ContextRunnable.run(ContextRunnable.java:37)
ServerImpl$JumpToApplicationThreadServerStreamListener$1HalfClosed.runInContext(ServerImpl.java:866)
ServerCallImpl$ServerStreamListenerImpl.halfClosed(ServerCallImpl.java:340)
TracingServerInterceptor$TracingServerCall$TracingServerCallListener.onHalfClose(TracingServerInterceptor.java:133)
Contexts$ContextualizedServerCallListener.onHalfClose(Contexts.java:86)
ForwardingServerCallListener$SimpleForwardingServerCallListener.onHalfClose(ForwardingServerCallListener.java:40)
ForwardingServerCallListener.onHalfClose(ForwardingServerCallListener.java:23)
PartialForwardingServerCallListener.onHalfClose(PartialForwardingServerCallListener.java:35)
ServerCalls$UnaryServerCallHandler$UnaryServerCallListener.onHalfClose(ServerCalls.java:182)
AdServiceGrpc$MethodHandlers.invoke(AdServiceGrpc.java:205)
AdService$AdServiceImpl.getAds(AdService.java:114)
AdService.access$300(AdService.java:41)
AdService.getAdsByCategory(AdService.java:140)
CopyrightCertification.certify(CopyrightCertification.java:32)
PhotographyCertifier.certify(PhotographyCertifier.java:16)
StockPhotos.isCopyright(StockPhotos.java:23)
StockPhotos$CopyrightPhoto.matchesAd(StockPhotos.java:59) StockPhotos...
StockPhotos.access$100(StockPhotos.java:13)
StockPhotos.createDatabase(StockPhotos.java:37)
StockPhotos$CopyrightPhoto.<init>(StockPhotos.java:54)

4

Call stack starts at beginning starting with frame 1

Type	Meaning of stack frame width
CPU	How often a function appears in stack traces. CPU usage relative to other stack traces.
Memory	How much memory is allocated by a function relative to other stack traces

SRE views the AlwaysOn Profiler - Memory

Sep 6, 2:41pm - 2:56pm | All Environments | Service: adservice | Host | Container | Process | Linked to spans | Clear All

AlwaysOn Profiling | adservice (Java) | CPU | Memory

1

Heap Memory

Heap Usage 98.5MB
Live Data Size 81.2MB
Max Heap Size 648.7MB

600MB, 400MB, 200MB, 0B
2:41:58 PM TUESDAY — 2:51:30 PM - 2:52:00 PM Tuesday Tuesday — 2:57:28 PM TUESDAY
Heap Usage | Live Data Size | Max Heap Size

Application Activity

App Active Time % 36%
App Paused for GC % 64%
Allocation Rate 0B/s

100%, 50%, 0%; 0B/s
2:41:58 PM TUESDAY — 2:51:30 PM - 2:52:00 PM Tuesday Tuesday — 2:57:28 PM TUESDAY
Allocation Rate | App Paused for GC % | App Active Time %

Garbage Collection Activity

Pause Duration 19.22s
Pause Count 622

600, 400, 200, 0; 18.54s, 12.36s, 6.18s, 0ms
2:41:58 PM TUESDAY — 2:51:30 PM - 2:52:00 PM Tuesday Tuesday — 2:57:28 PM TUESDAY
Pause Count | Pause Duration

2

Search stack frames

Using 554.3k call stacks out of 204.8k total

3

Name	Bytes Allocated	Count
hipstershop.copyright.StockPhotos$CopyrightPhoto.<init>(unknown:54)	3.4GB	531.9k
java.util.LinkedList.linkLast(unknown:142)	295.9KB	12.3k
hipstershop.copyright.StockPhotos.createDatabase(unknown:37)	280.7KB	553.5k
java.util.Arrays.copyOf(unknown:3332)	98.6KB	71
java.lang.StringCoding.encode(unknown:350)	79.1KB	59
java.util.Random.<init>(unknown:137)	61.2KB	2.5k
java.util.Arrays.copyOfRange(unknown:3664)	32.7KB	58
java.util.Arrays.copyOf(unknown:3236)	10.1KB	22
hipstershop.copyright.StockPhotos.createDatabase(unknown:35)	5.3KB	166
java.nio.DirectByteBuffer.slice(unknown:212)	3.2KB	50
io.netty.buffer.PooledUnsafeDirectByteBuf$1.newObject(unknown:34)	2.5KB	29
java.nio.DirectByteBuffer.duplicate(unknown:216)	1.7KB	27
io.netty.handler.codec.http2.DefaultHttp2FrameWriter.writeHeadersInternal(unknown:489)	1.5KB	19
java.util.ArrayDeque.<init>(unknown:193)	1.3KB	17

Memory Allocation Stack Traces

4

Thread.run(unknown:750)
ThreadPoolExecutor$Worker.run(unknown:624)
ThreadPoolExecutor.runWorker(unknown:1149)
SerializingExecutor.run(unknown:133)
ContextRunnable.run(unknown:37)
ServerImpl$JumpToApplicationThreadServerStreamListener$1HalfClosed.runInContext(unknown:866)
ServerCallImpl$ServerStreamListenerImpl.halfClosed(unknown:340)
TracingServerInterceptor$TracingServerCall$TracingServerCallListener.onHalfClose(unknown:133)
Contexts$ContextualizedServerCallListener.onHalfClose(unknown:86)
ForwardingServerCallListener$SimpleForwardingServerCallListener.onHalfClose(unknown:40)
ForwardingServerCallListener.onHalfClose(unknown:23)
PartialForwardingServerCallListener.onHalfClose(unknown:35)
ServerCalls$UnaryServerCallHandler$UnaryServerCallListener.onHalfClose(unknown:182)
AdServiceGrpc$MethodHandlers.invoke(unknown:205)
AdService$AdServiceImpl.getAds(unknown:114)
AdService.access$300(unknown:41)
AdService.getAdsByCategory(unknown:140)
CopyrightCertification.certify(unknown:32)
PhotographyCertifier.certify(unknown:16)
StockPhotos.isCopyright(unknown:23) Sto...
StockPhotos$CopyrightPhoto.matchesAd(unknown:59) Sto...
StockPhotos.access$100(unknown:13) Sto...
StockPhotos.createDatabase(unknown:37)
StockPhotos$CopyrightPhoto.<init>(unknown:54)

hipstershop.copyright.StockPhotos$CopyrightPhoto.(unknown:54)
94.8% | 2.9GB allocated

Copy Stack Trace

5

splunk> turn data into doing

Developer Picks up ticket for investigation

Triggered Major Alert

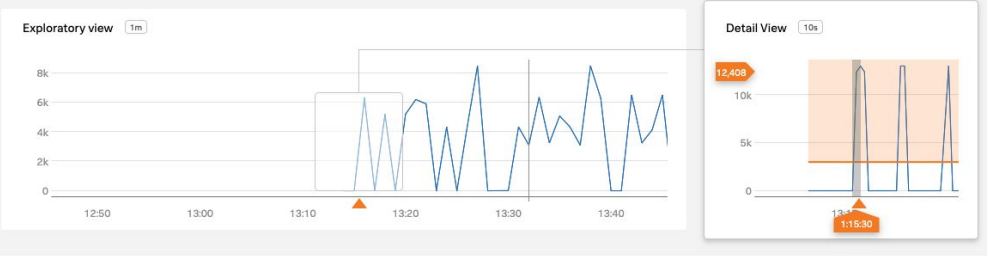
1

Request Latency distribution Detector

Alert Triggered 2 hours ago, on 08/19/2022 at 1:15:30 PM (UTC -05:00)

Time 08/19/2022 12:45:30 pm to 08/19/2022 01:45:30 pm

RUM Hipstershop Ad Service Get Ads Error Rate (value: 12,408) > 3000



Plots: d: metric_name (value: 12,408) Trigger Threshold (value: 3,000)

Show plot information

Message

Rule "Request Latency distribution Detector" in detector "OB:adservice:Request Latency Detector" triggered at Fri, 19 Aug 2022 18:15:30 GMT.

Triggering condition: The 90th percentile of latency is above 3000ms for 100% of 10s. Clears when below 1000ms for 100% of 1m.

Latency: 12407.975999999999ms Threshold: 3000ms Signal details: {sf_environment=vzmp-apm-env, sf_kind=SERVER, sf_operation=hipstershop.AdService/GetAds, sf_service=adservice}

About this Alert

Incident ID Fahgb3hA0AY

Next Steps

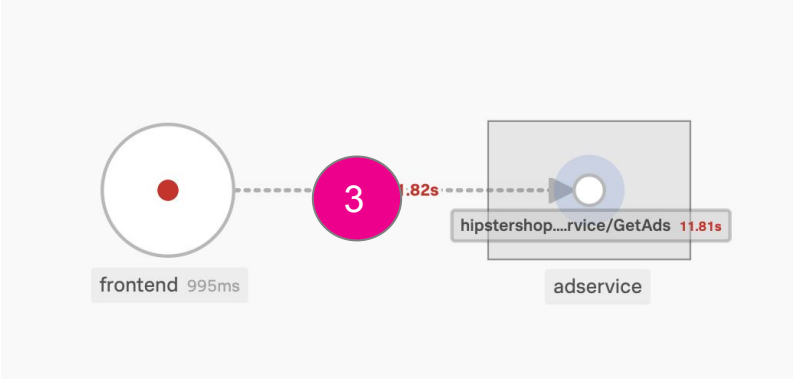
Tip *No tip has been set for this detector*

Runbook *No runbook has been set for this detector*

Explore Further

APM Troubleshoot

2

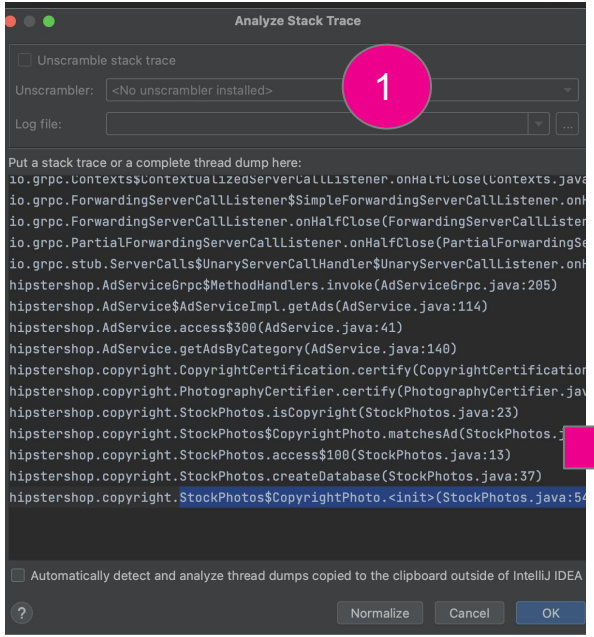


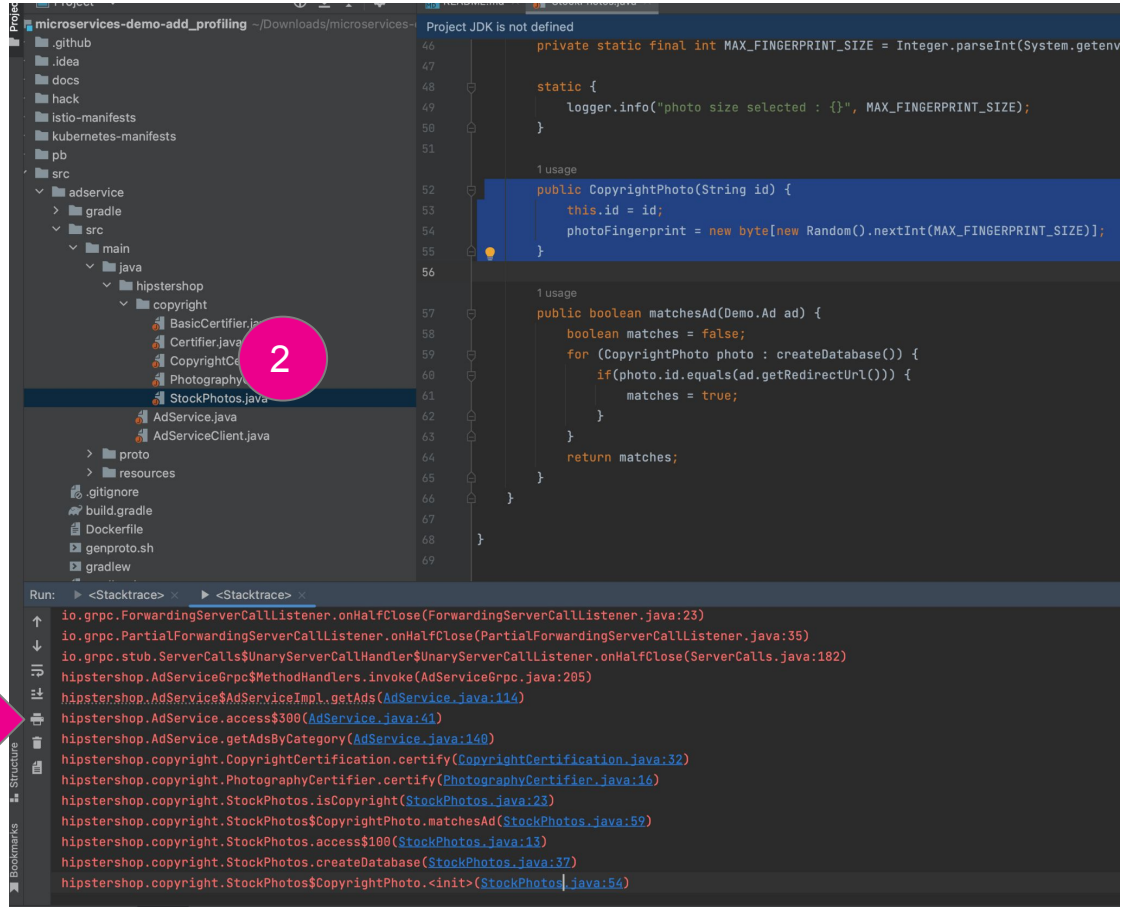
K8s cluster(s) for adservice

4

Logs for adservice in Observability

Developer analyzes code





In Summary - What did we do and learn

- We learned how to create effective detectors to alert not only on static thresholds but also on historical anomalies where normality is tested based upon historical data patterns. Drives proactive monitoring and observability.

- We demonstrated how an SRE can investigate in the context of the alert and identify a potential performance bottleneck by examining the slow traces, spans and operations. Followed by examination of the flame graph and capturing the call stack for the Jira ticket for developer engagement.

- We walked through how the developer can use the call stack captured and included in the SRE ticket to easily import into their IDE and navigate to the appropriate method highlighted by the call stack analysis.

- We can now apply these concepts to the CSCorp's DevOps processes to meet goals of improving performance and quality of the services deployed and also to move to a more proactive operating model. This results in business aligned value realization.

Thank You