Introdução à Observabilidade de Aplicações

O que é a Observabilidade de Aplicações?

Observabilidade de Aplicações permite entender um sistema externamente, possibilitando fazer perguntas sobre ele **sem precisar conhecer seu funcionamento interno**. Além disso, facilita a solução de problemas desconhecidos e imprevistos . A observabilidade ajuda a responder à pergunta: **“Por que isso está acontecendo?”**

Para fazer perguntas sobre um sistema, é necessário que sua aplicação esteja devidamente **instrumentada**. Isso significa que o código da aplicação ou algum componente intimamente ligado à aplicação como o Application Server, máquina virtual da linguagem da aplicação ou intepretador de comandos da linguagem devem emitir **sinais**, como **rastreamentos** (*traces*), **métricas** e **logs**. Uma aplicação está bem instrumentada quando os desenvolvedores não precisam adicionar mais instrumentação para investigar problemas, pois já possuem todas as informações necessárias.

O Priax, além de possuir mecanismos de instrumentação próprios pode é compatível com mecanismos consagrados do mercado como **OpenTelemetry, Jaeger ou Opensearch APM.** Todos esses mecanismos são usados para instrumentar o código da aplicação e tornar um sistema observável.

Telemetria, métricas e confiabilidade de Aplicações

Telemetria se refere aos dados emitidos por um sistema e seu comportamento. Esses dados podem ser apresentados na forma de **rastreamentos**, **métricas** e **logs**.

A **confiabilidade** responde à pergunta: **“O serviço está fazendo o que os usuários esperam que ele faça?”**. Por exemplo, um sistema pode estar disponível 100% do tempo, mas, se ao clicar em **"Adicionar ao Carrinho"** para incluir um par de sapatos pretos, o sistema não adicionar sempre os sapatos pretos, ele é considerado **não confiável**.

Métricas são agregações ao longo do tempo de dados numéricos sobre sua infraestrutura ou aplicação. Exemplos incluem:

- Taxa de erro do sistema
- Uso da CPU
- Taxa de solicitações de um serviço

SLI (*Service Level Indicator*) é uma medida do comportamento de um serviço. Um bom SLI mede o serviço **do ponto de vista dos usuários**. Um exemplo de SLI é a velocidade de carregamento de uma página web.

SLO (*Service Level Objective*) é a forma como a **confiabilidade** é comunicada dentro da organização ou para outras equipes, vinculando um ou mais SLIs ao **valor de negócio**.

Entendendo o Rastreamento Distribuído

O **rastreamento distribuído** permite observar solicitações conforme elas se propagam por sistemas distribuídos e complexos. Ele melhora a visibilidade sobre a saúde de uma aplicação ou sistema e ajuda a depurar comportamentos difíceis de reproduzir localmente.

O rastreamento distribuído é **essencial para sistemas distribuídos**, que frequentemente apresentam problemas não determinísticos ou complexos demais para serem reproduzidos em um ambiente local.

Para entender o rastreamento distribuído, é importante conhecer seus principais componentes: **logs**, **spans** e **traces**.

Logs

Um **log** é uma mensagem com carimbo de tempo emitida por serviços ou componentes. Diferente dos rastreamentos, os logs não estão necessariamente associados a uma solicitação ou transação específica. Eles estão presentes em praticamente todo software e foram amplamente utilizados por desenvolvedores e operadores para entender o comportamento dos sistemas.

Exemplo de log:

```
I, [2021-02-23T13:26:23.505892 #22473]  INFO -- : [6459ffe1-ea53-4044-aaa3-bf902868f730]
Started GET "/" for ::1 at 2021-02-23 13:26:23 -0800
```

No entanto, os logs **não são suficientes** para rastrear a execução do código, pois geralmente carecem de informações contextuais, como o local de origem da chamada.

Os logs se tornam muito mais úteis quando são incluídos como parte de um **span** ou quando estão correlacionados a um **trace** e um **span**.

Spans

Um **span** representa uma **unidade de trabalho** ou uma operação. Ele rastreia operações específicas de uma solicitação, fornecendo uma visão detalhada do que ocorreu durante sua

execução.

Um span inclui:

- **Nome**
- **Dados temporais**
- **Mensagens de log estruturadas**
- **Metadados (Atributos)** que fornecem mais informações sobre a operação rastreada.

Atributos de Span

Os **atributos** são metadados associados a um span.

Chave	Valor
http.request.method	GET
network.protocol.version	1.1
url.path	/webshop/articles/4
url.query	?s=1
server.address	example.com
server.port	8080
url.scheme	https
http.route	/webshop/articles/:article_id
http.response.status_code	200
client.address	192.0.2.4
client.socket.address	192.0.2.5 (o cliente passa por um proxy)
user_agent.original	Mozilla/5.0 (Windows NT 10.0; Win64; ...)

Rastreamentos Distribuídos

Um **rastreamento distribuído** (*trace*) registra os caminhos percorridos por solicitações (feitas por uma aplicação ou usuário final) enquanto atravessam **arquiteturas multi-serviço**, como aplicações baseadas em microsserviços ou serverless.

Um trace é composto por um ou mais **spans**:

- O **span raiz** representa o início e o fim de uma solicitação.
- Os spans filhos fornecem um **contexto detalhado** sobre o que ocorre durante a solicitação.

Sem rastreamento, identificar a causa raiz de problemas de desempenho em sistemas distribuídos pode ser desafiador. O rastreamento **simplifica a depuração** e facilita o entendimento de sistemas complexos, detalhando o que acontece com uma solicitação enquanto ela se propaga pelo sistema.

Muitos backends de observabilidade visualizam os **traces** como diagramas de cascata (*waterfall diagrams*), que demonstram a relação entre spans pai e filhos, representando relações **hierárquicas** e **aninhadas**.

Propagação de Contexto

Entenda o conceito que viabiliza o Rastreamento Distribuído.

Com a **propagação de contexto**, os **sinais** podem ser correlacionados entre si, independentemente de onde são gerados. Embora não se limite apenas ao rastreamento, a propagação de contexto permite que os **rastreamentos** construam informações causais sobre um sistema que está distribuído de forma arbitrária entre processos e limites de rede.

Para entender a propagação de contexto, é necessário conhecer dois conceitos principais: **contexto** e **propagação**.

Contexto

O **contexto** é um objeto que contém as informações necessárias para que os serviços emissores e receptores, ou unidades de execução, possam **correlacionar um sinal com outro**.

Por exemplo: se o **serviço A** chama o **serviço B**, um **span** do serviço A, cujo **ID** está presente no contexto, será utilizado como **span pai** para o próximo span criado no serviço B. O **ID do rastreamento** (*trace ID*) também será incluído no contexto e utilizado para o próximo span criado no serviço B. Isso significa que esse novo span fará parte do **mesmo rastreamento** que o span do serviço A.

Propagação

A **propagação** é o mecanismo responsável por **mover o contexto** entre serviços e processos. Ela **serializa** ou **desserializa** o objeto de contexto e fornece as informações relevantes para que o contexto seja transferido de um serviço para outro.

Geralmente, a propagação é gerenciada automaticamente pelas **bibliotecas de instrumentação** e é **transparente** para o usuário. No entanto, caso seja necessário realizar a propagação de contexto **manualmente**, você pode utilizar a **API de Propagadores** (*Propagators API*) de cada

biblioteca de instrumentação.

Revision #9
Created 2024-08-15 18:41:31 UTC by Wagner B. Simonato
Updated 2024-09-10 23:57:57 UTC by Wagner B. Simonato