浅谈skywalking的TraceSegmentServiceClient 原创

朱柿子 2020-03-23 23:12:53 ©著作权

文章标签 编程技术 java 面试 阅读数 1006

朱柿子
86 原创 4万 人气
0 翻译 0 转载
关注

近期文章
1.gRPC怎样节省您的开发
2.Html的css3法和python3
3.HeadFirst 解析Alibaba
4.HBase模式案例高_宽_
5.HBase模式案例研究列

热门文章
HeadFirst 解析Alibaba
gRPC怎样节省您的开发
HBase 支持的数据类型
Html的css3法和python3
Golang 才是学习指针的

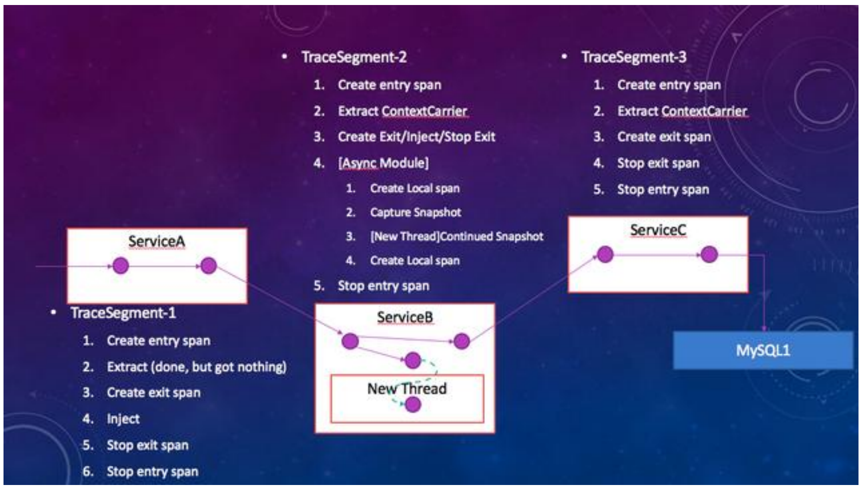
七日热门
RabbitMQ-进阶
Java开发工程师进阶篇-
性能环境之docker操作指
【8.30-9.5】上周精彩回
华为音频编辑服务带你一
Java（2）详解注释&关
【数据结构】——时间复
【分布式技术专题】带你
[C语言小白]简易扫雷程
Python中的函数参数：

相关标签
pinpoint skywalking
skywalking docker

本文参考原文-http://bjbsair.com/2020-03-22/tech-info/5102.html

序

本文主要研究一下skywalking的TraceSegmentServiceClient



TracingContextListener

skywalking-6.6.0/apm-sniffer/apm-agent-core/src/main/java/org/apache/skywalking/apm/agent/core/context/TracingContextListener.java

```java
public interface TracingContextListener {
    void afterFinished(TraceSegment traceSegment);
}
```

- TracingContextListener定义了afterFinished方法，其参数为TraceSegment

TraceSegment

skywalking-6.6.0/apm-sniffer/apm-agent-core/src/main/java/org/apache/skywalking/apm/agent/core/context/trace/TraceSegment.java

```java
public class TraceSegment {

    private ID traceSegmentId;

    private List<TraceSegmentRef> refs;

    private List<AbstractTracingSpan> spans;

    private DistributedTraceIds relatedGlobalTraces;

    private boolean ignore = false;

    private boolean isSizeLimited = false;

    private final long createTime;
```

```
        this.spans = new LinkedList<AbstractTracingSpan>();
        this.relatedGlobalTraces = new DistributedTraceIds();
        this.relatedGlobalTraces.append(new NewDistributedTraceId());
        this.createTime = System.currentTimeMillis();
    }

    public void ref(TraceSegmentRef refSegment) {
        if (refs == null) {
            refs = new LinkedList<TraceSegmentRef>();
        }
        if (!refs.contains(refSegment)) {
            refs.add(refSegment);
        }
    }

    public void relatedGlobalTraces(DistributedTraceId distributedTraceId) {
        relatedGlobalTraces.append(distributedTraceId);
    }

    public void archive(AbstractTracingSpan finishedSpan) {
        spans.add(finishedSpan);
    }

    public TraceSegment finish(boolean isSizeLimited) {
        this.isSizeLimited = isSizeLimited;
        return this;
    }

    public ID getTraceSegmentId() {
        return traceSegmentId;
    }

    public int getServiceId() {
        return RemoteDownstreamConfig.Agent.SERVICE_ID;
    }

    public boolean hasRef() {
        return !(refs == null || refs.size() == 0);
    }

    public List<TraceSegmentRef> getRefs() {
        return refs;
    }

    public List<DistributedTraceId> getRelatedGlobalTraces() {
        return relatedGlobalTraces.getRelatedGlobalTraces();
    }

    public boolean isSingleSpanSegment() {
        return this.spans != null && this.spans.size() == 1;
    }

    public boolean isIgnore() {
        return ignore;
    }

    public void setIgnore(boolean ignore) {
        this.ignore = ignore;
    }

    public UpstreamSegment transform() {
        UpstreamSegment.Builder upstreamBuilder = UpstreamSegment.newBuilder();
        for (DistributedTraceId distributedTraceId : getRelatedGlobalTraces()) {
            upstreamBuilder = upstreamBuilder.addGlobalTraceIds(distributedTraceId.toUniqueId());
        }
        SegmentObject.Builder traceSegmentBuilder = SegmentObject.newBuilder();
        /**
         * Trace Segment
```

skywalking elk

skywalking pinpoint

skywalking springclou

skywalking 告警

skywalking 报警

springcloud skywalkin

zipkin skywalking

文章目录

```java

        // SpanObject
        for (AbstractTracingSpan span : this.spans) {
            traceSegmentBuilder.addSpans(span.transform());
        }
        traceSegmentBuilder.setServiceId(RemoteDownstreamConfig.Agent.SERVICE_ID);
        traceSegmentBuilder.setServiceInstanceId(RemoteDownstreamConfig.Agent.SERVICE_INSTANCE_ID)
        traceSegmentBuilder.setIsSizeLimited(this.isSizeLimited);

        upstreamBuilder.setSegment(traceSegmentBuilder.build().toByteString());
        return upstreamBuilder.build();
    }

    @Override
    public String toString() {
        return "TraceSegment{" +
            "traceSegmentId='" + traceSegmentId + '\'' +
            ", refs=" + refs +
            ", spans=" + spans +
            ", relatedGlobalTraces=" + relatedGlobalTraces +
            '}';
    }

    public int getApplicationInstanceId() {
        return RemoteDownstreamConfig.Agent.SERVICE_INSTANCE_ID;
    }

    public long createTime() {
        return this.createTime;
    }
}
```

- TraceSegment定义了traceSegmentId、refs、spans、relatedGlobalTraces等属性；它提供了ref、relatedGlobalTraces、archive 、finish、transform等方法

IConsumer

skywalking-6.6.0/apm-commons/apm-datacarrier/src/main/java/org/apache/skywalking/apm/commons/datacarrier/consumer/IConsumer.java

```java
public interface IConsumer<T> {
    void init();

    void consume(List<T> data);

    void onError(List<T> data, Throwable t);

    void onExit();
}
```

- IConsumer定义了init、consume、onError、onExit方法

TraceSegmentServiceClient

skywalking-6.6.0/apm-sniffer/apm-agent-core/src/main/java/org/apache/skywalking/apm/agent/core/remote/TraceSegmentServiceClient.java

```java
@DefaultImplementor
public class TraceSegmentServiceClient implements BootService, IConsumer<TraceSegment>, TracingCo
    private static final ILog logger = LogManager.getLogger(TraceSegmentServiceClient.class);
    private static final int TIMEOUT = 30 * 1000;

    private long lastLogTime;
    private long segmentUplinkedCounter;
    private volatile GRPCChannelStatus status = GRPCChannelStatus.DISCONNECT;

    @Override
    public void prepare() throws Throwable {
        ServiceManager.INSTANCE.findService(GRPCChannelManager.class).addChannelListener(this);
    }

    @Override
    public void boot() throws Throwable {
        lastLogTime = System.currentTimeMillis();
        segmentUplinkedCounter = 0;
        segmentAbandonedCounter = 0;
        carrier = new DataCarrier<TraceSegment>(CHANNEL_SIZE, BUFFER_SIZE);
        carrier.setBufferStrategy(BufferStrategy.IF_POSSIBLE);
        carrier.consume(this, 1);
    }

    @Override
    public void onComplete() throws Throwable {
        TracingContext.ListenerManager.add(this);
    }

    @Override
    public void shutdown() throws Throwable {
        TracingContext.ListenerManager.remove(this);
        carrier.shutdownConsumers();
    }

    @Override
    public void init() {

    }

    @Override
    public void consume(List<TraceSegment> data) {
        if (CONNECTED.equals(status)) {
            final GRPCStreamServiceStatus status = new GRPCStreamServiceStatus(false);
            StreamObserver<UpstreamSegment> upstreamSegmentStreamObserver = serviceStub.withDeadl
                @Override
                public void onNext(Commands commands) {
                    ServiceManager.INSTANCE.findService(CommandService.class).receiveCommand(comma
                }

                @Override
                public void onError(Throwable throwable) {
                    status.finished();
                    if (logger.isErrorEnable()) {
                        logger.error(throwable, "Send UpstreamSegment to collector fail with a grp
                    }
                    ServiceManager.INSTANCE.findService(GRPCChannelManager.class).reportError(thro
                }

                @Override
                public void onCompleted() {
                    status.finished();
                }
            });

            try {
                for (TraceSegment segment : data) {
                    UpstreamSegment upstreamSegment = segment.transform();
                    upstreamSegmentStreamObserver.onNext(upstreamSegment);
                }
            } catch (Throwable t) {
                logger.error(t, "Transform and send UpstreamSegment to collector fail.");
            }

            upstreamSegmentStreamObserver.onCompleted();
        } else {
            segmentAbandonedCounter += data.size();
        }

        printUplinkStatus();
    }

    private void printUplinkStatus() {
        long currentTimeMillis = System.currentTimeMillis();
        if (currentTimeMillis - lastLogTime > 30 * 1000) {
            lastLogTime = currentTimeMillis;
            if (segmentUplinkedCounter > 0) {
                logger.debug("{} trace segments have been sent to collector.", segmentUplinkedCour
                segmentUplinkedCounter = 0;
            }
            if (segmentAbandonedCounter > 0) {
                logger.debug("{} trace segments have been abandoned, cause by no available channel
                segmentAbandonedCounter = 0;
            }
        }
    }

    @Override
    public void onError(List<TraceSegment> data, Throwable t) {
        logger.error(t, "Try to send {} trace segments to collector, with unexpected exception.",
    }

    @Override
    public void onExit() {

    }

    @Override
    public void afterFinished(TraceSegment traceSegment) {
        if (traceSegment.isIgnore()) {
            return;
        }
        if (!carrier.produce(traceSegment)) {
            if (logger.isDebugEnable()) {
                logger.debug("One trace segment has been abandoned, cause by buffer is full.");
            }
        }
    }

    @Override
    public void statusChanged(GRPCChannelStatus status) {
        if (CONNECTED.equals(status)) {
            Channel channel = ServiceManager.INSTANCE.findService(GRPCChannelManager.class).getCha
            serviceStub = TraceSegmentReportServiceGrpc.newStub(channel);
        }
        this.status = status;
    }
}
```

- TraceSegmentServiceClient实现了BootService、IConsumer、TracingContextListener、GRPCChannelListener接口；其prepare方法往GRPCChannelManager注册自身的channelListener；其boot方法设置lastLogTime，实例化DataCarrier，并设置其consumer为自身；其onComplete方法执行TracingContext.ListenerManager.add(this)；其shutdown方法执行TracingContext.ListenerManager.remove(this)以及carrier.shutdownConsumers()；其consume方法在status为CONNECTED的时候执行upstreamSegmentStreamObserver.onNext(upstreamSegment)、upstreamSegmentStreamObserver.onCompleted()以及status.wait4Finish()；其afterFinished方法执行carrier.produce(traceSegment)；其statusChanged设置serviceStub及status

ConsumerThread

```
public class ConsumerThread<T> extends Thread {
    private volatile boolean running;
    private IConsumer<T> consumer;
    private List<DataSource> dataSources;
    private long consumeCycle;

    ConsumerThread(String threadName, IConsumer<T> consumer, long consumeCycle) {
        super(threadName);
        this.consumer = consumer;
        running = false;
        dataSources = new ArrayList<DataSource>(1);
        this.consumeCycle = consumeCycle;
    }

    /**
     * add whole buffer to consume
     *
     * @param sourceBuffer
     */
    void addDataSource(QueueBuffer<T> sourceBuffer) {
        this.dataSources.add(new DataSource(sourceBuffer));
    }

    @Override
    public void run() {
        running = true;

        final List<T> consumeList = new ArrayList<T>(1500);
        while (running) {
            if (!consume(consumeList)) {
                try {
                    Thread.sleep(consumeCycle);
                } catch (InterruptedException e) {
                }
            }
        }

        // consumer thread is going to stop
        // consume the last time
        consume(consumeList);

        consumer.onExit();
    }

    private boolean consume(List<T> consumeList) {
        for (DataSource dataSource : dataSources) {
            dataSource.obtain(consumeList);
        }

        if (!consumeList.isEmpty()) {
            try {
                consumer.consume(consumeList);
            } catch (Throwable t) {
                consumer.onError(consumeList, t);
            } finally {
                consumeList.clear();
            }
            return true;
        }
        return false;
    }

    void shutdown() {
        running = false;
    }

    /**
```

```
        private QueueBuffer<T> sourceBuffer;

        DataSource(QueueBuffer<T> sourceBuffer) {
            this.sourceBuffer = sourceBuffer;
        }

        void obtain(List<T> consumeList) {
            sourceBuffer.obtain(consumeList);
        }
    }
}
```

- ConsumerThread继承了Thread，其run方法会循环执行consume(consumeList)，跳出循环时会再次执行consume(consumeList)，最后执行consumer.onExit()；consume方法会遍历dataSources，执行其dataSource.obtain(consumeList)，然后在consumeList不为空的时候执行consumer.consume(consumeList)方法

ConsumeDriver

skywalking-6.6.0/apm-commons/apm-datacarrier/src/main/java/org/apache/skywalking/apm/commons/datacarrier/consumer/ConsumeDriver.java

```
public class ConsumeDriver<T> implements IDriver {
    private boolean running;
    private ConsumerThread[] consumerThreads;
    private Channels<T> channels;
    private ReentrantLock lock;

    public ConsumeDriver(String name, Channels<T> channels, Class<? extends IConsumer<T>> consumer
        long consumeCycle) {
        this(channels, num);
        for (int i = 0; i < num; i++) {
            consumerThreads[i] = new ConsumerThread("DataCarrier." + name + ".Consumser." + i + "
            consumerThreads[i].setDaemon(true);
        }
    }

    public ConsumeDriver(String name, Channels<T> channels, IConsumer<T> prototype, int num, long
        this(channels, num);
        prototype.init();
        for (int i = 0; i < num; i++) {
            consumerThreads[i] = new ConsumerThread("DataCarrier." + name + ".Consumser." + i + "
            consumerThreads[i].setDaemon(true);
        }

    }

    private ConsumeDriver(Channels<T> channels, int num) {
        running = false;
        this.channels = channels;
        consumerThreads = new ConsumerThread[num];
        lock = new ReentrantLock();
    }

    private IConsumer<T> getNewConsumerInstance(Class<? extends IConsumer<T>> consumerClass) {
        try {
            IConsumer<T> inst = consumerClass.newInstance();
            inst.init();
            return inst;
        } catch (InstantiationException e) {
            throw new ConsumerCannotBeCreatedException(e);
        } catch (IllegalAccessException e) {
            throw new ConsumerCannotBeCreatedException(e);
        }
    }

    @Override
```

```java
            }
            try {
                lock.lock();
                this.allocateBuffer2Thread();
                for (ConsumerThread consumerThread : consumerThreads) {
                    consumerThread.start();
                }
                running = true;
            } finally {
                lock.unlock();
            }
        }

        @Override
        public boolean isRunning(Channels channels) {
            return running;
        }

        private void allocateBuffer2Thread() {
            int channelSize = this.channels.getChannelSize();
            /**
             * if consumerThreads.length < channelSize
             * each consumer will process several channels.
             *
             * if consumerThreads.length == channelSize
             * each consumer will process one channel.
             *
             * if consumerThreads.length > channelSize
             * there will be some threads do nothing.
             */
            for (int channelIndex = 0; channelIndex < channelSize; channelIndex++) {
                int consumerIndex = channelIndex % consumerThreads.length;
                consumerThreads[consumerIndex].addDataSource(channels.getBuffer(channelIndex));
            }

        }

        @Override
        public void close(Channels channels) {
            try {
                lock.lock();
                this.running = false;
                for (ConsumerThread consumerThread : consumerThreads) {
                    consumerThread.shutdown();
                }
            } finally {
                lock.unlock();
            }
        }
    }
```

- ConsumeDriver实现了IDriver接口，其ConsumeDriver会创建num个ConsumerThread；其begin方法会执行allocateBuffer2Thread，给每个consumerThread添加dataSource，然后执行consumerThread.start()；其close方法会执行consumerThread.shutdown()

小结

TraceSegmentServiceClient实现了BootService、IConsumer、TracingContextListener、GRPCChannelListener接口；其prepare方法往GRPCChannelManager注册自身的channelListener；其boot方法设置lastLogTime，实例化DataCarrier，并设置其consumer为自身；其onComplete方法执行TracingContext.ListenerManager.add(this)；其shutdown方法执行TracingContext.ListenerManager.remove(this)以及carrier.shutdownConsumers()；其consume方法在status为CONNECTED的时候执行upstreamSegmentStreamObserver.onNext(upstreamSegment)、upstreamSegmentStreamObserver.onCompleted()以及status.wait4Finish()；其afterFinished方法执行carrier.produce(traceSegment)；其statusChanged设置serviceStub及status

赞 收藏 评论 分享

上一篇：帮你搞懂Python进程,线程与协程

下一篇：浅谈skywalking的spring-webflux-plugin

提问和评论都可以，用心的回复会被更多人看到 评论

相关文章

skywalking install

本文以8.1.0为例 下载地址 http://skywalking.apache.org/downloads/ 解压 配置 config目录 配置文件 config/applic…

skywalking安装

1、下载skywalking 下载地址1：https://skywalking.apache.org/downloads/ 自行选择ES6还是ES7版本的，或者使…

skywalking部署

【Skywalking基本介绍】点击【下载】到官方站点下载需要的软件版本，推荐下载二进制安装包项目地址：https://github.com/ap…

Skywalking系列博客2-Skywalking使用

本文探讨如何使用Skywalking监控应用。 Skywalking有多种使用方式，目前最流行(也是最强大)的使用方式是基…

SkyWalking链路监控（一）：SkyWalking快速搭建

简介 当分布式系统服务比较多，特别是微服务，出现故障就很难排查。所以需要借助APM 系统进行排查（Applic…

SkyWalking快速接入

任何技术和理念都将不能成为解决一切问题的银弹，有的只是权衡和选择"点击上方蓝色字体，关注我 在上一篇 …

springboot使用skywalking

skywalking安装参照本篇文章 javascript:void(0) 复制agent文件 从skywalking的agent目录，复制到本地某个目录…

helm部署SkyWalking

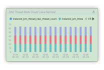
克隆chart到本地 git clone https://github.com/apache/skywalking-kubernetes cd skywalking-kubernetes/chart helm repo add elas…

SkyWalking 简单使用

Apache SkyWalking 分布式系统的应用程序性能监视工具，专为微服务、云原生架构和基于容器（Docker、K8s、Mesos）架构…

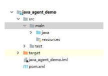
Skywalking-05：在Skywalking RocketBot上添加监控图表

在 Skywalking RocketBot 上添加监控图表 效果图 该图的一些配置信息如下： 标题为： JVM Thread State Count …



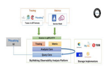
快速学习-Skywalking原理

4.Skywalking原理 4.1 java agent原理 上文中我们知道，要使用Skywalking去监控服务，需要在其 VM 参数中添加 ...



快速学习-skywalking入门

对于APM不了解的同学请先查看上一篇skywalking概述 1.2 什么是Skywalking 1.2.1 Skywalking概述 根据官方的解...

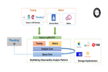


APM系统SkyWalking介绍

公司最近在构建服务化平台，需要上线APM系统，本篇文章简单的介绍SkyWalking APM APM全称Application Performance Ma...

skywalking搭建与使用

前言 在分布式环境中，对于服务的监控与链路追踪变得越来越重要，简单来说，相比单体应用，分布式环境下的...



Skywalking部署及使用

Skywalking部署及使用前言首先有必要说明一下为什么使用skywalking。我对zipkin、cat和skywalking这几个较为...

如何开启Apache SkyWalking的自监控?

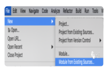
1. 开启Prometheus遥测数据默认情况下, 遥测功能（telemetry）是关闭的(selector 为 none)，像这样：telemetry: ...

Zipkin之外的选择：Skywalking vs Pinpoint

说明：本次对比基于skywalking-6.0.0-GA和Pinpoint-1.8.2（截止2019-02-19最新版本）。另外，我们这次技术选型直接否定...

Skywalking系列博客6-手把手教你编写 Skywalking 插件

在正式进入编写环节之前，建议先花一点时间了解下javaagent（这是JDK 5引入的一个玩意儿，最好了解下其工...



51CTO博客 技术成就梦想

友情链接

51CTO鸿蒙社区 51CTO学堂
51CTO

关于我们

官方博客 意见反馈 了解我们 全
在线客服 博客问答 网站地图 热