使用Apache SkyWalking APM 监控 Apache ServiceComb

吴晟 Sheng Wu
Huawei DevCloud

http://skywalking.io
Twitter @AsfSkyWalking

个人介绍

* GitHub： https://github.com/wu-sheng
* Personal Homepage： https://wu-sheng.github.io/me/

* Apache SkyWalking creator, PPMC member and Committer
* 2018 Microsoft MVP
* OpenTracing OTSC & OTAIB member
* W3C Trace Context specification member
* 2017 GSoC(Google Summer of Code) mentor

SkyWalking 2015 - 2018

2015

- 创建项目
- 提供SDK通过手工埋点的方式抓取链路信息，链路的spanId的生成规则参考阿里的鹰眼

2016

- 使用byte-buddy实现自动埋点
- 自动埋点的实现采用插件的机制区分不同的框架和不同的版本
- 探针重构：优化性能，移除鹰眼的spanId的生成规则

2017

- 探针与后端的协议重构：为后端分析而生
- 后端重构：提供链路数据的分析和分类聚合
- 界面重构：提供一个APM系统最基本的能力（应用拓扑图、链路展示、服务关系依赖、JVM统计指标）

2018

- 界面重构：比肩甚至超越商业APM系统的界面
- 探针：其他语言的支持（Php，Pthon，Go...）
- 后端：通过GraphQL将前后端接口协议化，更丰富的统计指标、预警、历史数据自动清理、路由节点推测，提升性能和稳定性

QCon 全球软件开发大会

演讲：SkyWalking的发展之路——从无名小卒到拥抱全球

地点：第二会议厅C

所属专题：崛起的中国开源软件市场

中国开源项目的全球化，起步难，而且苦难重重。在OSS这个领域，国内更是很少涉猎，那么SkyWalking是怎么样从第一行代码，慢慢成长为如今的Apache孵化器项目，从CNCF OpenTracing到W3C trace context标准，都有SkyWalking的身影，在风风的美国之行中，更是得到Google、Microsoft、Zipkin、AppDynamic、New Relic、DynaTrace、Instana等一系列世界顶级公司的尊重与重视，还荣获本次Workshop的热门分享话题，以及MVP，

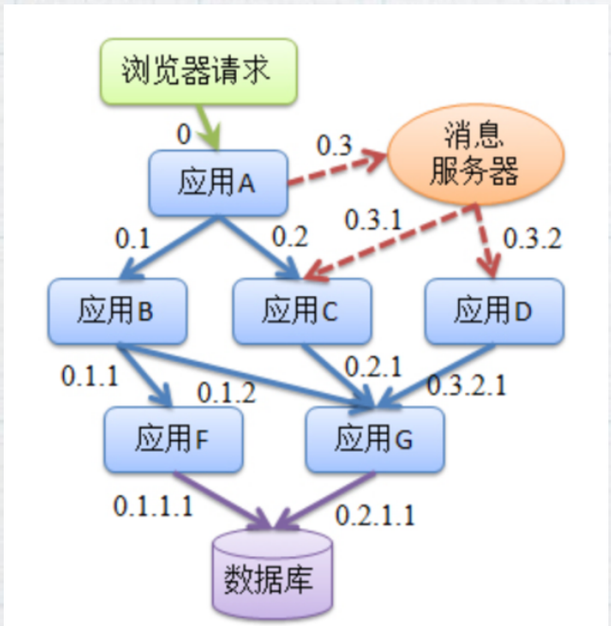
理论与原理

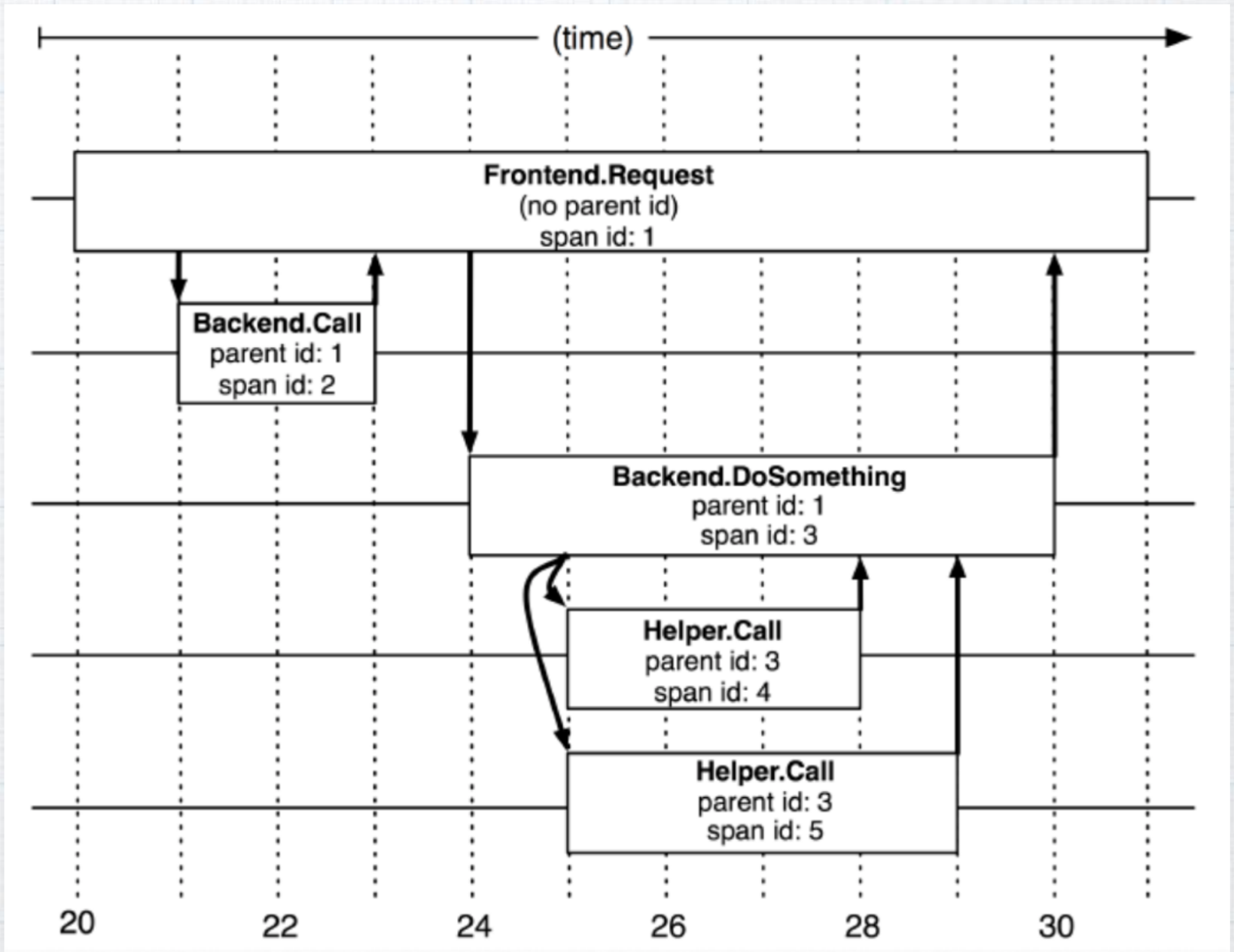
理论基础：Google Dapper Paper

鹰眼 != Dapper

It is just an implementation

绝大多数的追踪系统，
并不使用这种编号方式





探针工作原理

Running with Instrumentation, Pseudocode Only

```
PreparedStatement updateSales = con.prepareStatement(
          "UPDATE COFFEES SETSALES = ? WHERE COF_NAME LIKE ? ");
updateSales.setInt(1, 75);
updateSales.setString(2, "Colombian");
updateSales.executeUpdate();
```

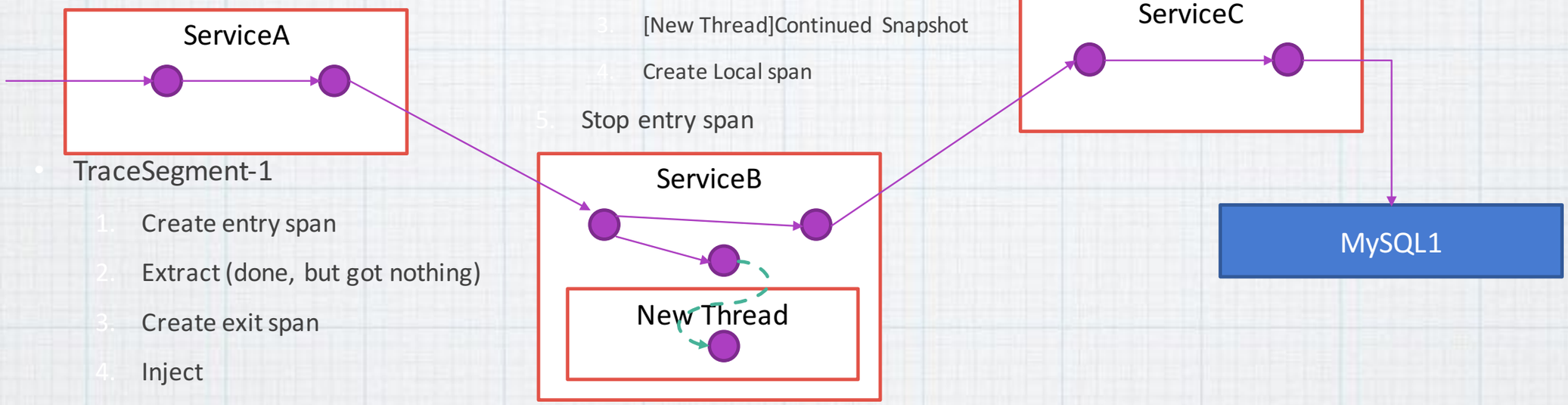
Auto-instrumentation mechanism
AOP(Aspect Oriented Programming) without Spring Framework

```
PreparedStatement updateSales = con.prepareStatement(
          "UPDATE COFFEES SETSALES = ? WHERE COF_NAME LIKE ? ");
tracer.cacheSQL("UPDATE COFFEES SETSALES = ? WHERE COF_NAME LIKE ? ");
updateSales.setInt(1, 75);
tracer.cacheDbParam(1, 75);
updateSales.setString(2, "Colombian");
tracer.cacheDbParam(2, "Colombian");
Span span = tracer.createSpan().start();
updateSales.executeUpdate();
span.tag(Tags.Basic.TYPE, "Database");
span.tag(Tags.SQL, tracer.getCachedSQL());
span.tag(Tags.SQL_PARAMETERS, tracer.getCachedDbParams());
span.stop();
```

- TraceSegment-1
  1. Create entry span
  2. Extract (done, but got nothing)
  3. Create exit span
  4. Inject
  5. Stop exit span
  6. Stop entry span

- TraceSegment-2
  1. Create entry span
  2. Extract ContextCarrier
  3. Create Exit/Inject/Stop Exit
  4. [Async Module]
     1. Create Local span
     2. Capture Snapshot
     3. [New Thread]Continued Snapshot
     4. Create Local span
  5. Stop entry span

- TraceSegment-3
  1. Create entry span
  2. Extract ContextCarrier
  3. Create exit span
  4. Stop exit span
  5. Stop entry span



分布式追踪原理

Plugin Mechanism

Instrumentation Define

```java
public class RealCallInstrumentation extends ClassInstanceMethodsEnhancePluginDefine {

    private static final String ENHANCE_CLASS = "okhttp3.RealCall";

    private static final String INTERCEPT_CLASS = "org.apache.skywalking.apm.plugin.okhttp.v3.RealCallInterceptor";

    @Override protected ClassMatch enhanceClass() { return NameMatch.byName(ENHANCE_CLASS); }

    @Override protected ConstructorInterceptPoint[] getConstructorsInterceptPoints() {
        return new ConstructorInterceptPoint[] {
            new ConstructorInterceptPoint() {
                @Override public ElementMatcher<MethodDescription> getConstructorMatcher() { return any(); }
                @Override public String getConstructorInterceptor() { return INTERCEPT_CLASS; }
            }
        };
    }

    @Override protected InstanceMethodsInterceptPoint[] getInstanceMethodsInterceptPoints() {
        return new InstanceMethodsInterceptPoint[] {
            new InstanceMethodsInterceptPoint() {
                @Override public ElementMatcher<MethodDescription> getMethodsMatcher() { return named("execute"); }
                @Override public String getMethodsInterceptor() { return INTERCEPT_CLASS; }
                @Override public boolean isOverrideArgs() { return false; }
            },
            new InstanceMethodsInterceptPoint() {
                @Override public ElementMatcher<MethodDescription> getMethodsMatcher() {
                    return named("enqueue").and(takesArguments( length: 1));
                }
                @Override public String getMethodsInterceptor() {
                    return "org.apache.skywalking.apm.plugin.okhttp.v3.EnqueueInterceptor";
                }
                @Override public boolean isOverrideArgs() { return false; }
            }
        };
    }
```

需要增强的类

插码的实现类

定义拦截点：构造函数

匹配所有构造

定义拦截点：方法

匹配名称为execute的方法

匹配类名

Before method

```java
@Override public void beforeMethod(EnhancedInstance objInst, Method method, Object[] allArguments,
    Class<?>[] argumentsTypes, MethodInterceptResult result) throws Throwable {
    Request request = (Request)objInst.getSkyWalkingDynamicField();

    ContextCarrier contextCarrier = new ContextCarrier();
    HttpUrl requestUrl = request.url();
    AbstractSpan span = ContextManager.createExitSpan(requestUrl.uri().getPath(), contextCarrier, remotePeer: requestUrl.host() + ":" + requestUrl.port());
    span.setComponent(ComponentsDefine.OKHTTP);
    Tags.HTTP.METHOD.set(span, request.method());
    Tags.URL.set(span, requestUrl.uri().toString());
    SpanLayer.asHttp(span);

    Field headersField = Request.class.getDeclaredField( name: "headers");
    Field modifiersField = Field.class.getDeclaredField( name: "modifiers");
    modifiersField.setAccessible(true);
    modifiersField.setInt(headersField, i: headersField.getModifiers() & ~Modifier.FINAL);

    headersField.setAccessible(true);
    Headers.Builder headerBuilder = request.headers().newBuilder();
    CarrierItem next = contextCarrier.items();
    while (next.hasNext()) {
        next = next.next();
        headerBuilder.add(next.getHeadKey(), next.getHeadValue());
    }
    headersField.set(request, headerBuilder.build());
}
```

创建Exit Span，设置关键属性

设置其他属性：

1. 组件类型
2. 所属层
3. 非重要Tag和Log

传递上下文

After method

```
@Override
public Object afterMethod(EnhancedInstance objInst, Method method, Object[] allArguments,
    Class<?>[] argumentsTypes, Object ret) throws Throwable {
    Response response = (Response)ret;
    int statusCode = response.code();

    AbstractSpan span = ContextManager.activeSpan();
    if (statusCode >= 400) {
        span.errorOccurred();
        Tags.STATUS_CODE.set(span, Integer.toString(statusCode));
    }

    ContextManager.stopSpan();

    return ret;
}

@Override public void handleMethodException(EnhancedInstance objInst, Method method, Object[] allArguments,
    Class<?>[] argumentsTypes, Throwable t) {
    AbstractSpan abstractSpan = ContextManager.activeSpan();
    abstractSpan.errorOccurred();
    abstractSpan.log(t);
}
```

Key documents

原理资料

* Dapper论文：https://research.google.com/pubs/pub36356.html

* OpenTracing官网：http://opentracing.io/

* OpenTracing中文：https://github.com/opentracing-contrib/opentracing-specification-zh

重点资料地址

* GitHub: https://github.com/apache/incubator-skywalking
* 安装： https://github.com/apache/incubator-skywalking/blob/master/docs/cn/Quick-start-CN.md
* 支持的插件列表： https://github.com/apache/incubator-skywalking/blob/master/docs/Supported-list.md
* 编译工程： https://github.com/apache/incubator-skywalking/blob/master/docs/cn/How-to-build-CN.md

ServiceComb 1.0.0 under SkyWalking 5.0.0 monitoring

127.0.0.1:8080/#/dashboard

Sky Walking

Last 15 minutes

- Dashboard
- Topology
- Application
- Server
- Service
- Trace
- Alarm

App 4

Service 9

DB & Cache 1

MQ 0

Avg Application Alarm 0.00%

Max 0%
Min 0%

H2 | 100.00% | 1

Slow Service

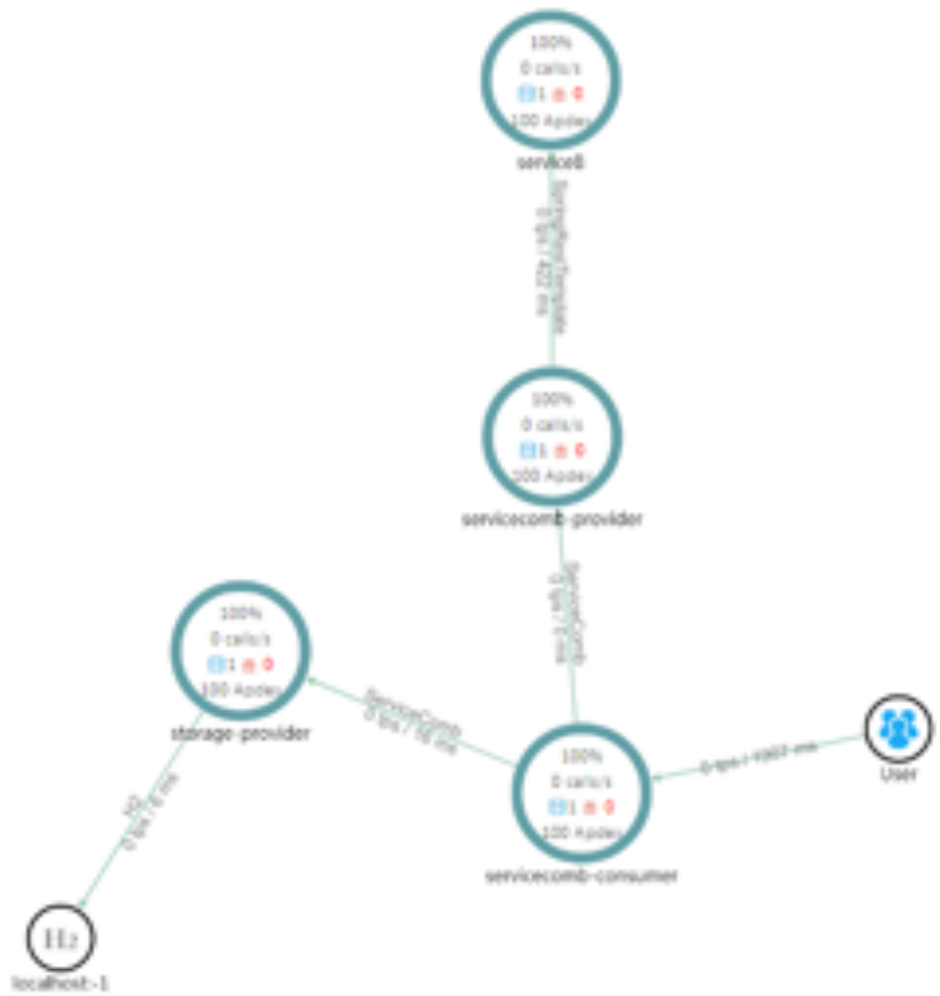
1. codefirst.codeFirstSpringmvcHello.sayHi — 0 ms
2. codefirstClient.codeFirstSpringmvcHello.say — 1907 ms
3. catchH2.catchH2.sayHi — 531 ms
4. /serviceB/rest — 203 ms
5. codefirst.codeFirstJaxrsHello.sayHi — 562 ms

Application Throughput

1. storage-provider — 0 t/s
2. serviceB — 0 t/s
3. servicecomb-provider — 0 t/s
4. servicecomb-consumer — 0 t/s

SkyWalking GitHub

Topolgy Map



11232@szxg9002818651:servicecomb-consumer

Info

OS: servicecomb-consumer_10.57.202.192

Host Name: szxg9002818651

Process Id: 11232

IPv4: 10.57.202.192

Avg Response Time

122.81 ms

Avg Calls Per Second

0 ms

CPU %

Heap MB

Non-Heap MB

ice

codefirstClient.codeFirstSpringmvcHello.say

Avg Throughout

0.00

Avg Response Time

124.06 ms

Avg SLA

100 %

Dependency Map

SkyWalking GitHub

http://49.4.12.44:8080/#/dashboard

国内在线demo环境

信息发布和讨论组

* 官方信息发布：https://twitter.com/AsfSkyWalking
* QQ（中文）：392443393
* Gitter（英文）：https://gitter.im/openskywalking/Lobby
* 问题提交与需求讨论：https://github.com/apache/incubator-skywalking/issues

Thanks