容器环境下基于APM的海量日志全链路跟踪分析

谐云科技副总裁 茨程

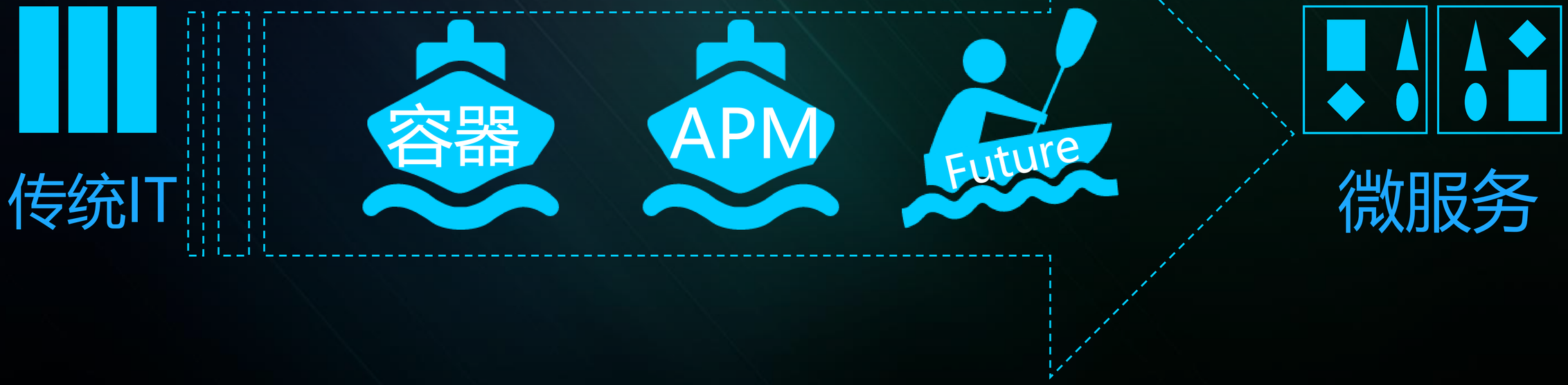
讲师介绍

芪程

谐云科技联合创始人，专注于分布式系统容错，大数据处理和分析，主导设计了谐云APM产品。

未来的IT架构一定是微服务化的

容器支持服务，APM定位故障

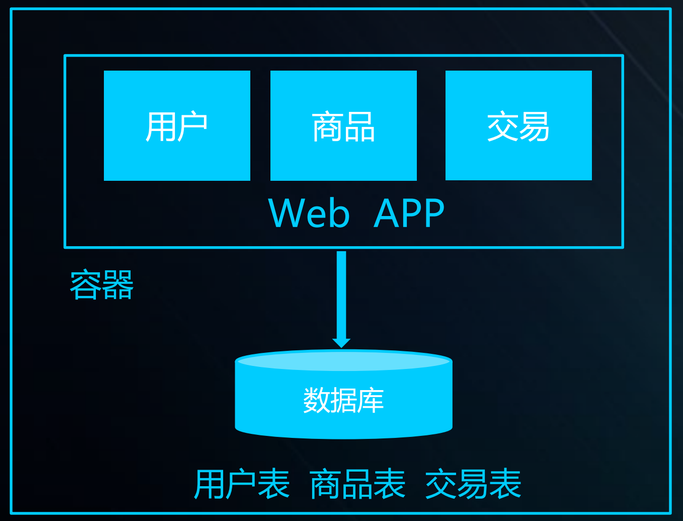


谐云拥有这两款产品，且有很好结合点

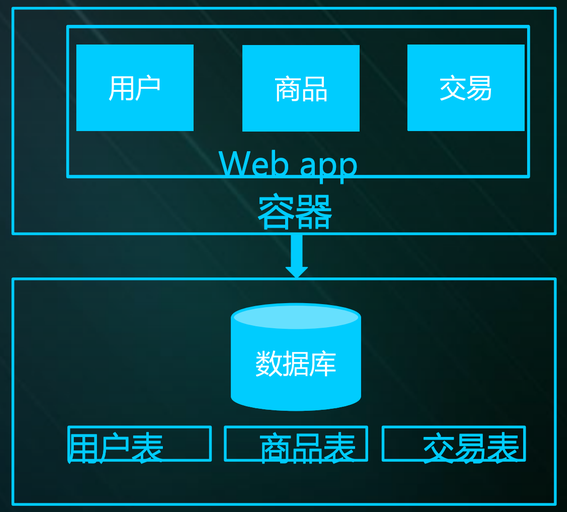
目录

CONTENTS

简单应用架构的日志排查过程



单台机器



应用服务器

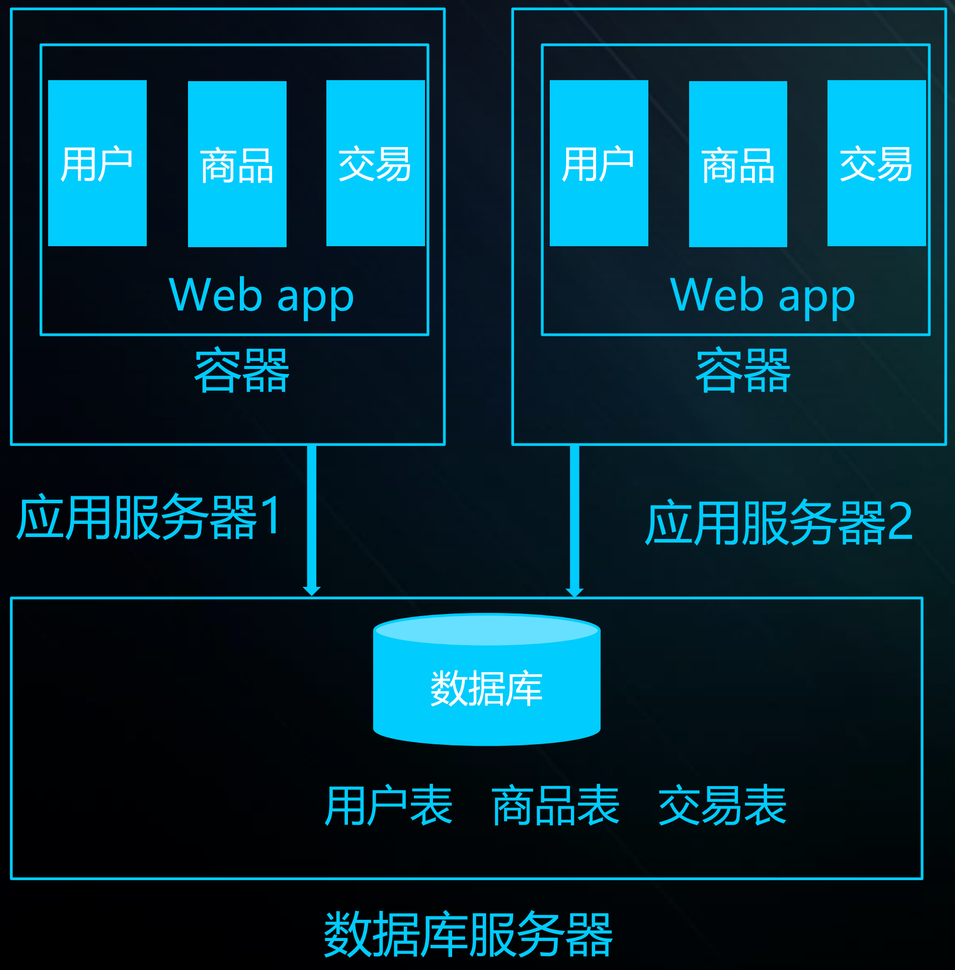
数据库服务器

当应用架构较简单的时候

只有一个日志文件，直接查看文件即可

EASY

稍微复杂一点应用架构的排查过程

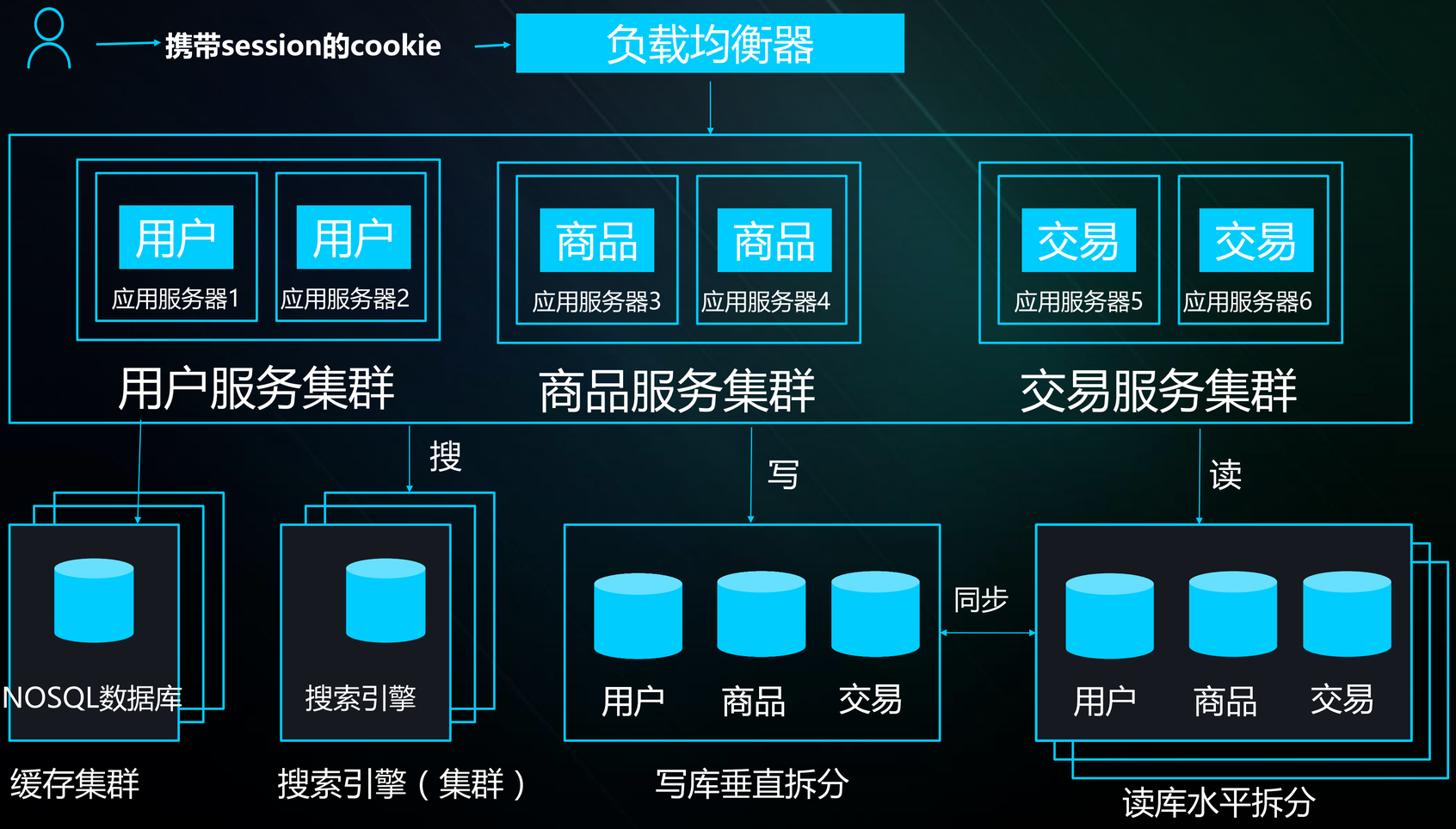


一个集群里面有限的几个应用日志

多花点时间，直接查看文件即可

有点烦，但还能接受

较复杂的应用架构的日志排查过程

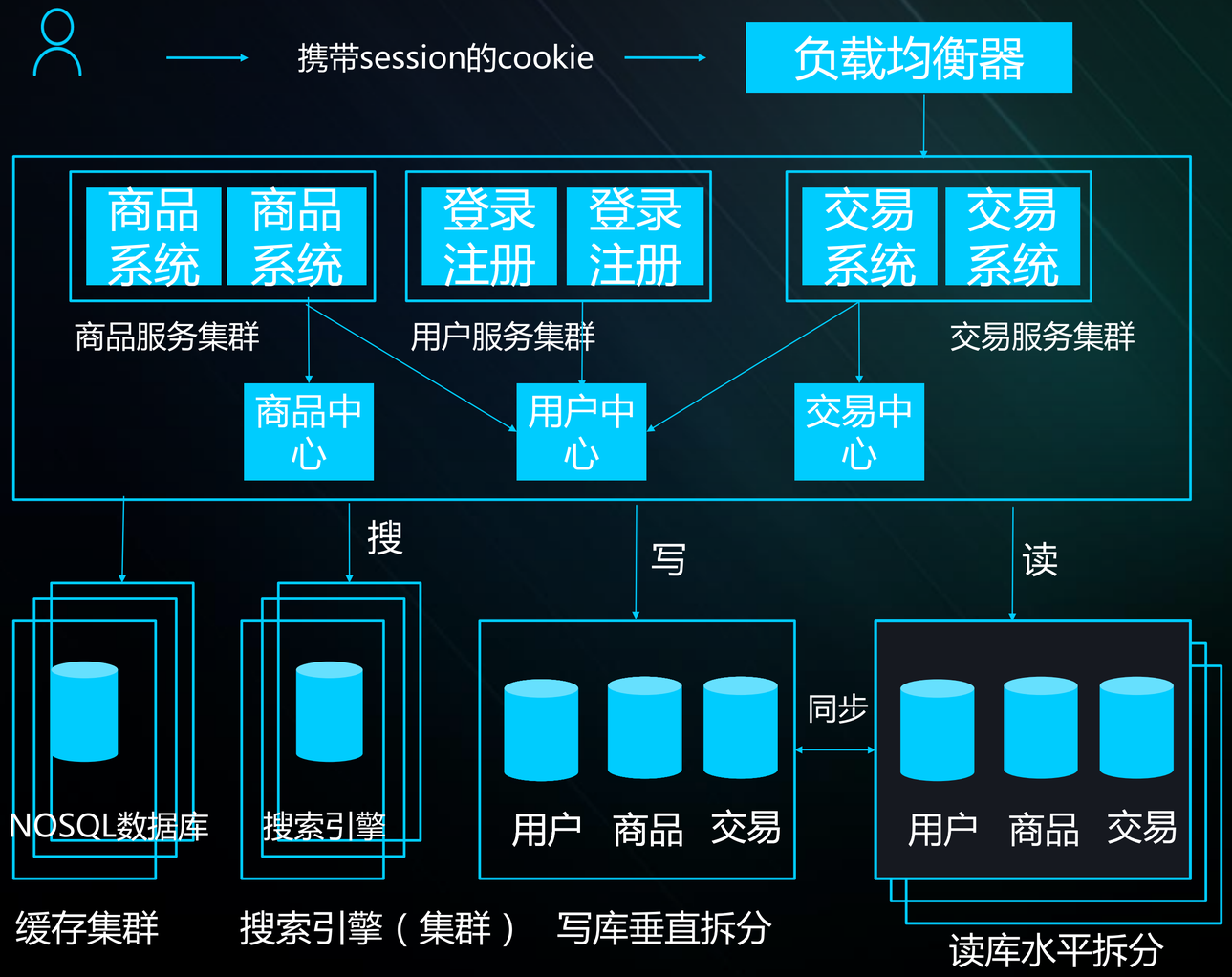


基于ELK的统一日志

- 搜索
- 文件导出

在统一日志支撑下，还能统一接受

典型基于微服务调用的日志排查过程



容器环境的特点

1. 弹性伸缩

2. 相同的服务实例数很多几十个，甚至上百个

统一日志为啥不能完全解决问题？

1 统一日志的使用方式是：搜索

2 在微服务的整条链路上的应用都需要关心

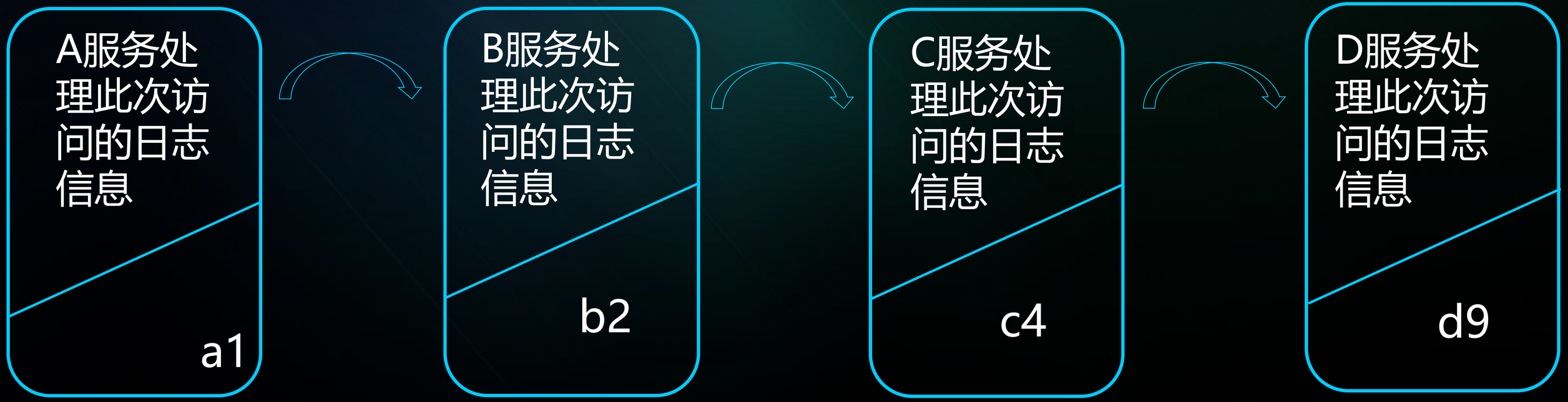
3 结论：搜索出来的非相关性信息太多

- 比如搜索：Error，那意味整个系统所有的微服务的Error日志可能都需要看
- 运气好找到一个相关性的日志之后，再去关联其它服务节点上的日志很复杂，无从下手

全链路的日志含义

可以将用户一次访问相关的日志信息经过所有服务不同实例完全找出来，并且能够按照实际响应顺序显示出来

Mar 02, 2016 10:12:28 AM org.apache.coyote.http11.Http11Protocol pause
INFO: Pausing Coyote HTTP/1.1 on http-8080
Mar 02, 2016 10:12:29 AM org.apache.catalina.core.StandardService stop
INFO: Stopping service Catalina
Mar 02, 2016 10:12:29 AM org.apache.coyote.http11.Http11Protocol destroy
INFO: Stopping Coyote HTTP/1.1 on http-8080

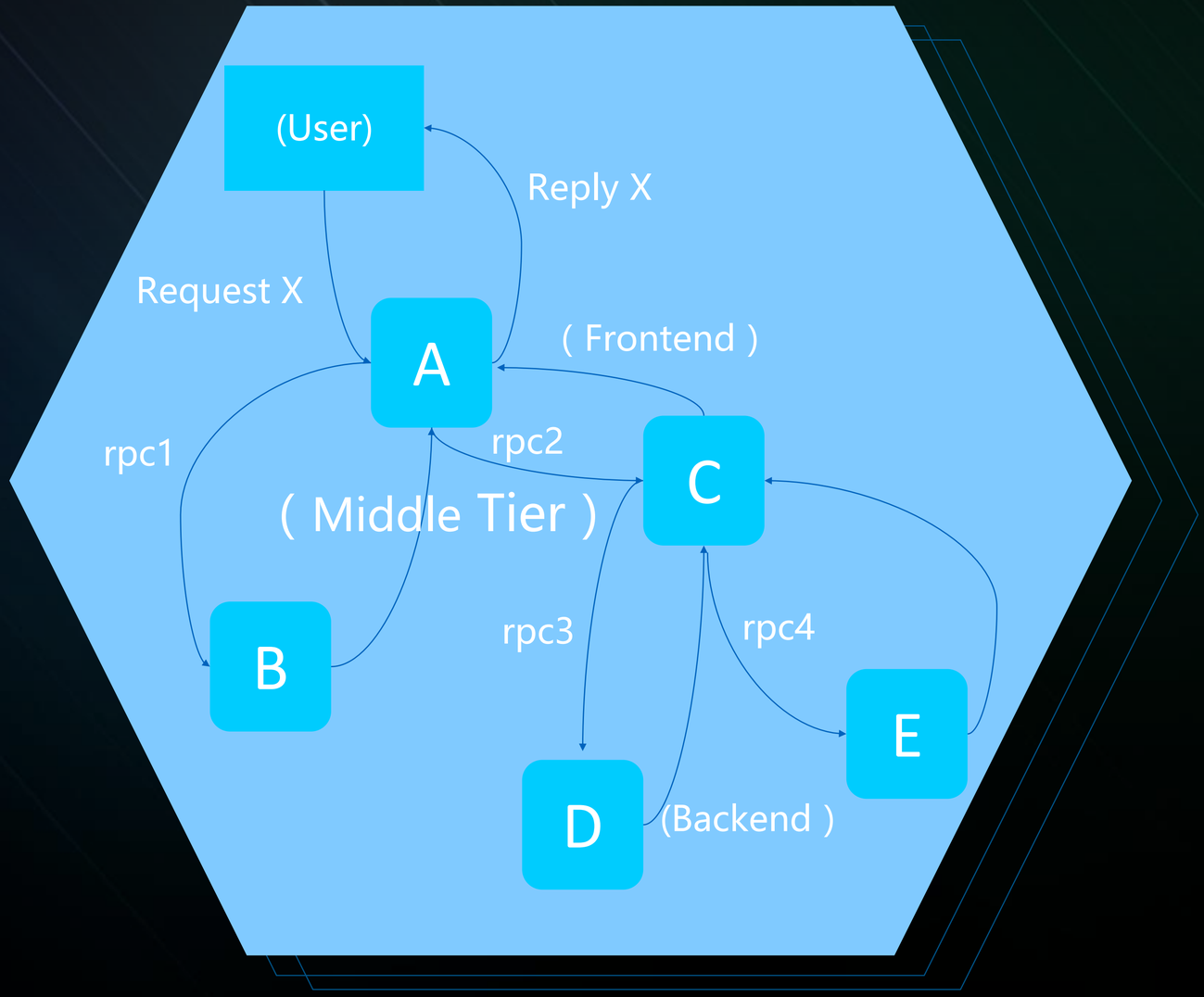


目录

CONTENTS

- 为什么需要全链路的日志排查
- 全链路的日志排查
- 代码中如何实现全链路
- 如何基于代码增强的方式实现日志增强
- 容器云带来的挑战

什么是全链路？Google Dapper的全链路



为什么要全链路？

快速交付与快速排查

VIP用户：张总，想买**50万**的理财产品失败，登陆不上。。。

老板：李导，我马上联系相关人，周末请你喝酒（抱拳状）

老板：50万的理财产品登陆不上！火速解决！！！

前台：我没问题，问问服务报错了吗？

中台：是你没调过来吧，我连日志都没打！

后台：胡说什么？我这有反应，你传错了！

老板：50万的理财产品登陆不上！火速解决！！！

运维：进程都在，也没看见ZB报错呀！

DBA：叫什么叫，连数据都没连上！

基础：傻X,磁盘满了，都不知道在叫啥。。。

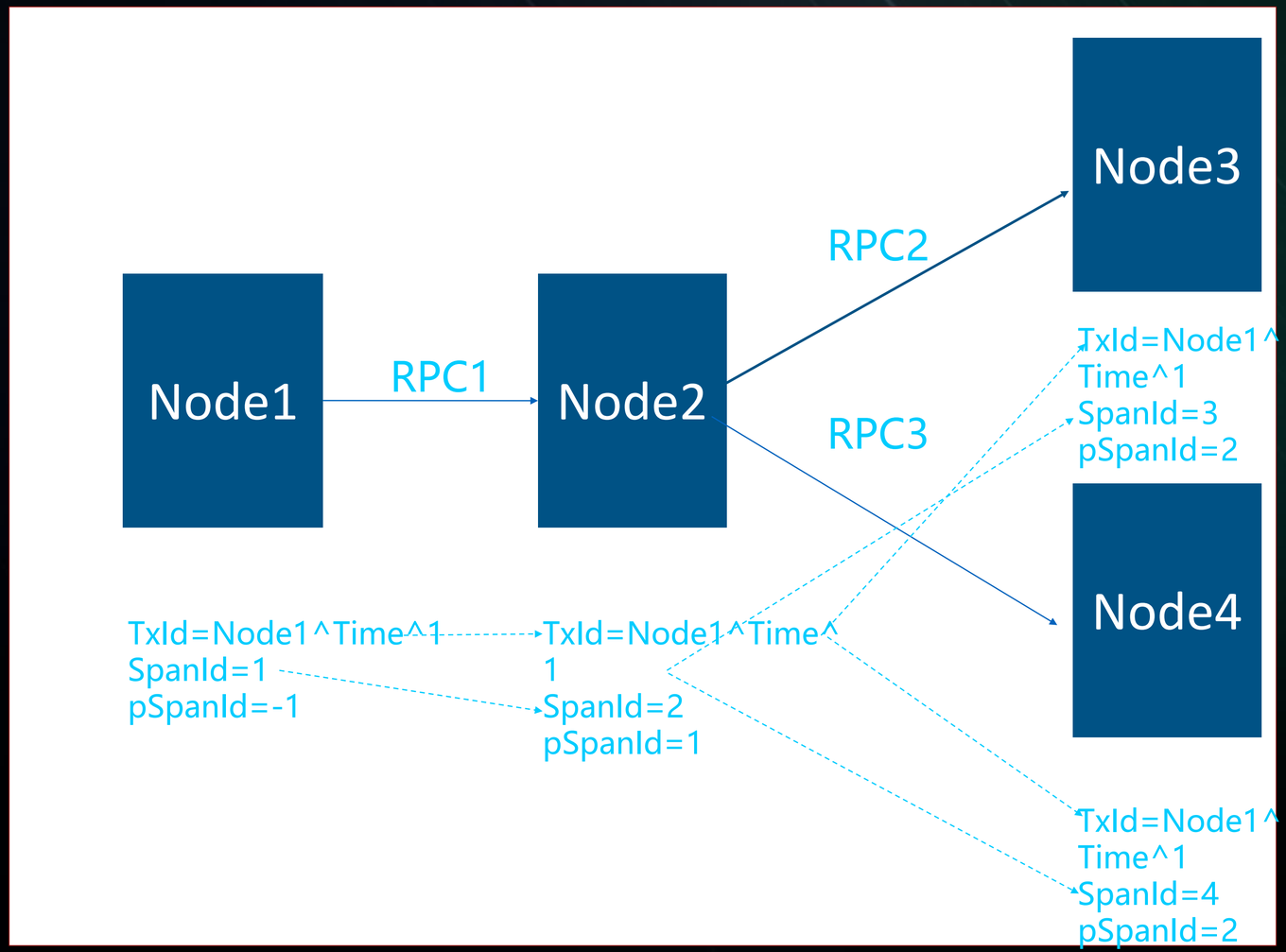
老板：张三说李四，李四说王五，说了半天都两个小时了（叹气状），就不能解决地快点吗？

员工：谁不想快啊？但也要快的起来呀！老板，我们都没偷懒啊（无奈状）。。。。。。

目录

CONTENTS

全链路的实现



TraceID：识别用户一次请求，所有全链路上的节点共用一个TraceID

SpanID：正在处理用户请求的节点

ParentSpanID：正在处理用户请求节点的上一个节点

Opentracing VS 自动化插装

Opentracing的优略势

①
适用于任何代码框架

②
对代码开发团队需要有较强的要求，测试团队需要配合一起测试

VS

自动化插装的优劣势

①
无需代码开发

②
支持的框架有限制

目录

CONTENTS

全链路日志示例

[2017-12-06 18:53:36] [ERROR] (AccountAction:206) - AccountAction.prepare... , transactionId = test@jeeshopclient-3177904429-875h5:8080^1512552181462^74

[2017-12-06 18:53:36] [ERROR] (AccountAction:678) - toLogin... , transactionId = test@jeeshopclient-3177904429-875h5:8080^1512552181462^74

[2017-12-06 18:53:36] [ERROR] (FrontInterceptor:28) - CommonInterceptor.intercept... , transactionId = test@jeeshopclient-3177904429-875h5:8080^1512552181462^74

Java常用的日志框架配置

Log4j
Log4j2
Logback

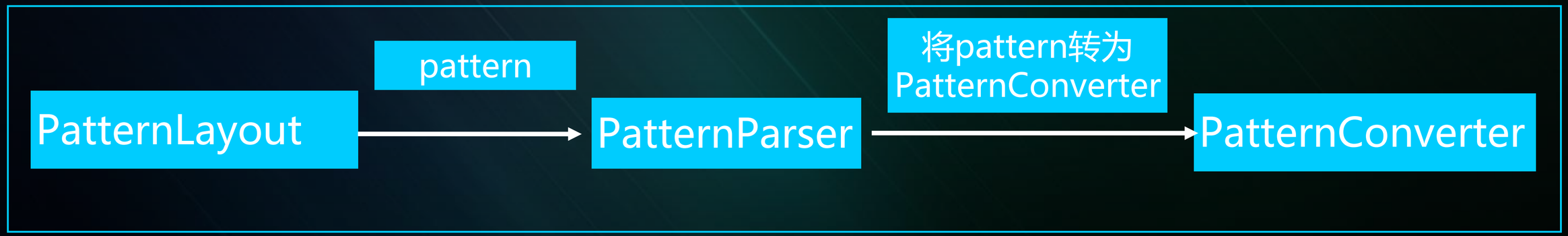
- 日志配置基本相似
- 配置日志格式化输出方式（Layout）
- 配置日志模版（Pattern）
- Layout的主要作用就是解析Pattern，格式化输出日志信息

日志常见配置（Log4J为例）

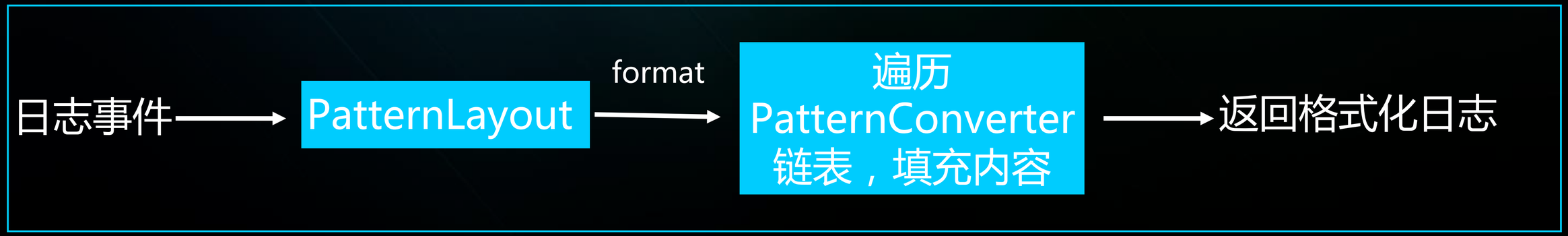
配置日志输出方式	log4j.appender.stdout=org.apache.log4j.ConsoleAppender
配置日志路径	log4j.appender.stdout.Target=System.out
配置日志格式化方式	log4j.appender.stdout.layout=org.apache.log4j.PatternLayo
配置具体格式，通过layout解析pattern	log4j.appender.stdout.layout.ConversionPattern =%-d{yyyy-MM-dd HH:mm:ss} %m%n

日志解析过程

- 根据配置文件，实例化PatternLayout，解析pattern
PatternConverter对象实质就是一个链表，把pattern中的参数拆解



- 每当日志事件到来，调用PatternLayout的format获取格式化日志



日志增强输出全链路的traceid

自定义开发PatternLayout、PatternParser、PatternConverter

在配置文件中需要将layout参数指定为自定义的PatternLayout

```
log4j.appender.CONSOLE.layout=org.xx.TracePatternLayout

log4j.appender.CONSOLE.layout.ConversionPattern=%d [%T] %-5p %c{1}:%L - %m%n
```

修改pattern，添加%T。“%T”是能够被PatternConverter识别的，可以将其转化为TraceID

基于APM日志增强方式

无需额外配置就可以开启日志增强

拦截某方法，追加Pattern格式，加入%T（只执行一次）

通过每次打印日志，都会调用format方法，按照pattern打印具体信息。通过拦截format，注入%T的值。

目录

CONTENTS

Kubernetes 中IP不固定带来的问题

IP不再能唯一的标识一个应用了

10：00 AM ：实际产生日志时的链路信息
10.10.101.100(A应用)->10.10.101.105(B应用)-> 10.10.101.119(C应用
)

13：00 PM：分析日志时，日志所记录的IP对应的服务已经不再是实际产生日志的服务
10.10.101.100(C应用)->10.10.101.105(D应用)-> 10.10.101.119(E应用)

如何解决？

How?

优点

实现简单

方案一

记录日志时，同时多记录一个字段，对应的是应用名称

缺点

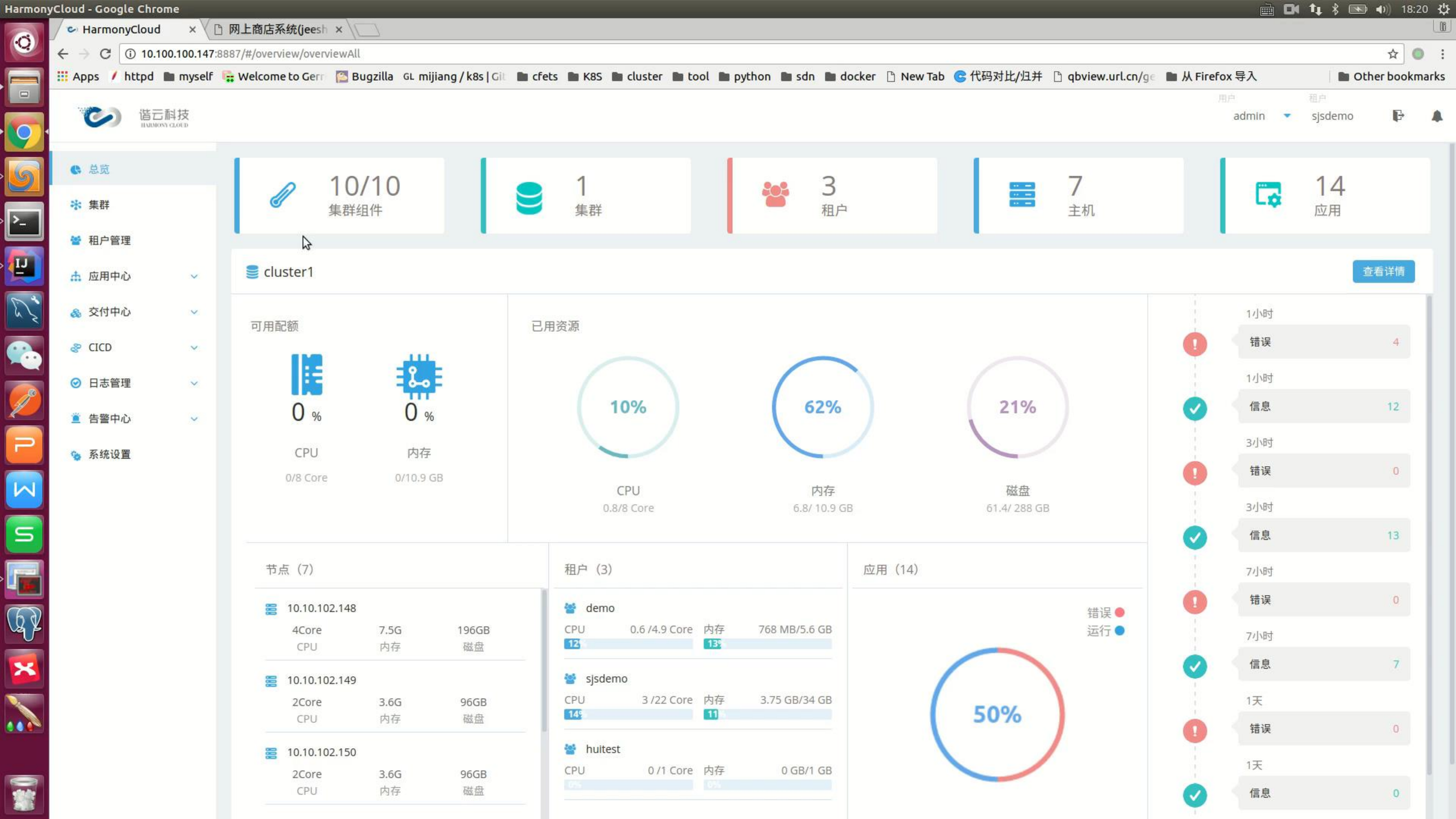
浪费存储

通过Pod Name来唯一的标识应用

采集日志数据时，按照pod name进行应用标识

APM的agent也按照pod name进行应用标识

HarmonyCloud - Google Chrome



微信

芪程


CNTC
SEL
Software Engineering Lab
Zhejiang University
感谢聆听
Thanks！