



# 微服务架构电商系统下的故障识别与分类

战队名：JustDo

答辩选手：贺健

公司：中兴通讯

2022 CCF国际AIOps挑战赛决赛暨AIOps研讨会



# CONTENTS

## 目录

第一节 团队介绍

第二节 赛题分析

第三节 解决之道

第四章节 总结展望



AIOps  
Challenge

2022 CCF国际AIOps挑战赛决赛  
暨AIOps研讨会

# 第一章节

# 团队介绍

# 团队介绍



参赛队名: JustDo

参赛选手: 贺健、谢勤政

团队简介:

JustDo战队成员来自于中兴通讯网络智能化研发中心ZXVMAX-AI项目。ZXVMAX-AI致力于建设自智网络、服务数智化等方面的一站式AI应用平台。AIOps技术是其中关键的组成部分，是ZXVMAX-AI网络智能化生态中不可或缺的一环。目前ZXVMAX-AI项目已经在自研AI平台上推出多种基于AIOps技术的应用或能力，助力客户建设高等级的自动驾驶网络。



AIOps  
Challenge

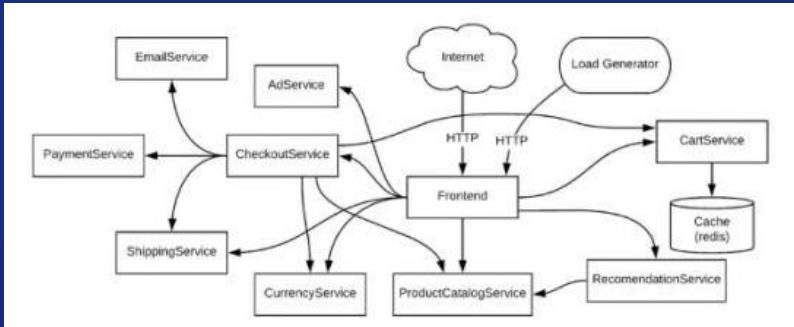
2022 CCF国际AIOps挑战赛决赛  
暨AIOps研讨会

## 第二章节

# 赛题分析

# 赛题分析

## □ 问题描述



- service服务：
    - 10个Service: emailservice, adservice, cartservice, shippingservice, productcatalogservice, currencyservice, recommendationservice, paymentservice, checkoutservice, frontend。
    - 每个Service下有4个Pod: emailservice-0, emailservice-1, emailservice-2, emailservice2-0等。
  - node虚机
    - 6个Node。
    - Pod都是动态地部署在Node上。

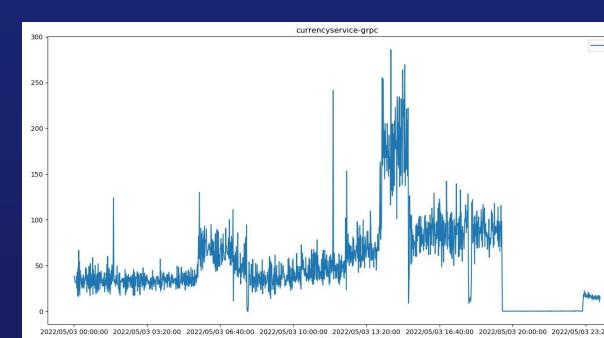
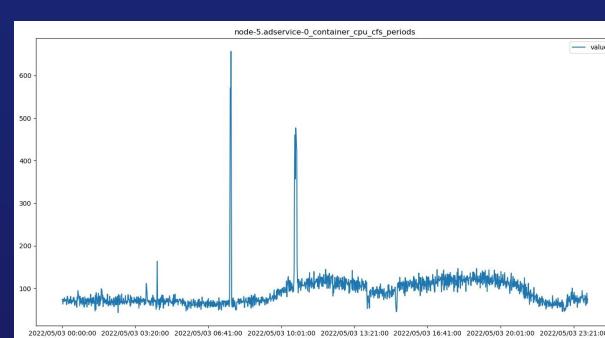
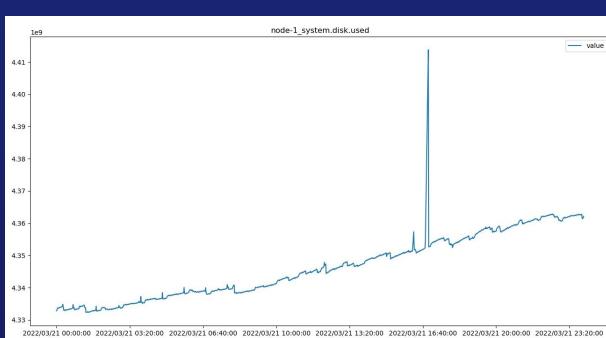
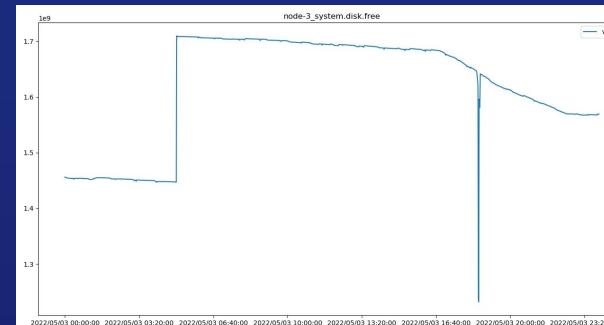
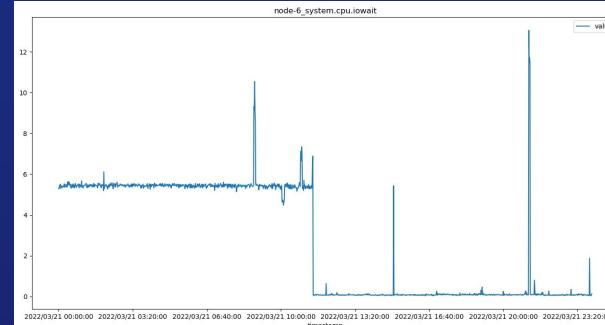
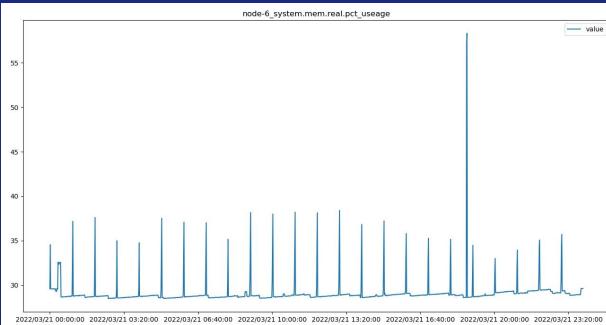
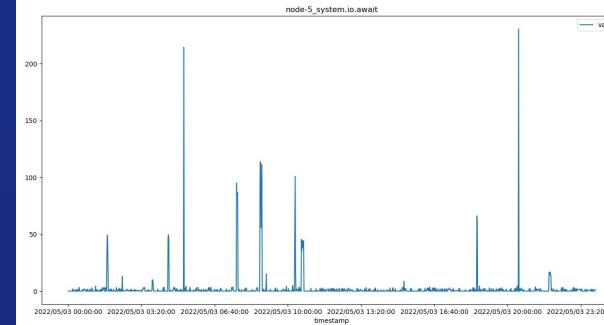
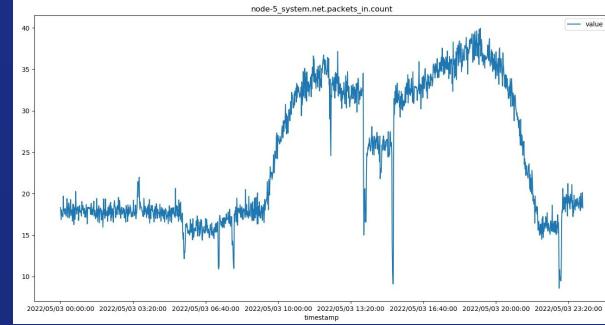
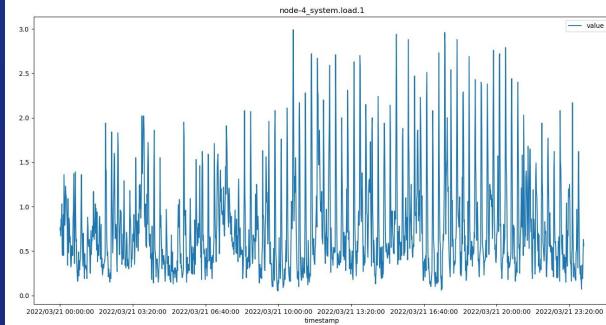
- 问题建模 (故障发现 --> 故障定位 --> 故障分类)

### 三步走：

- 故障发现：实时监测系统，及时发现故障
  - 故障定位：定界发生故障的根因故障对象
    - 3个层级：service级别（10个）、pod级别（40个）、node级别（6个）
  - 故障分类：定位根因故障对象的故障类型
    - service和Pod级别的故障主要有9种，如下图。
    - node的故障类型主要有6种，如下图。

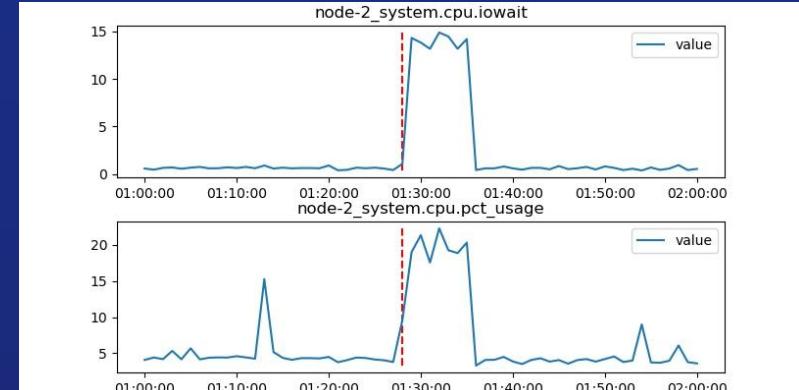
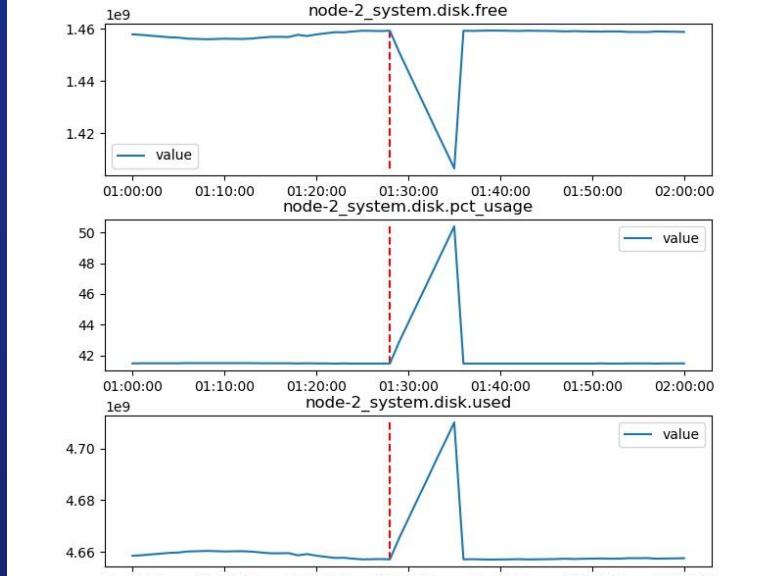


## □ 赛题挑战一：复杂多变的时序指标

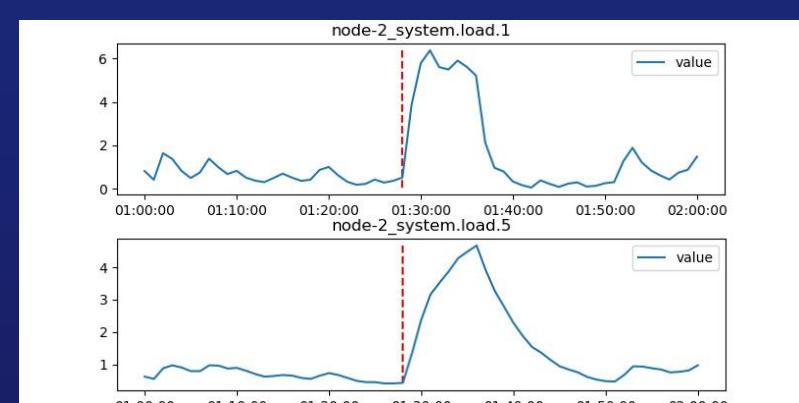


## □ 赛题挑战二：严重的伴生异常

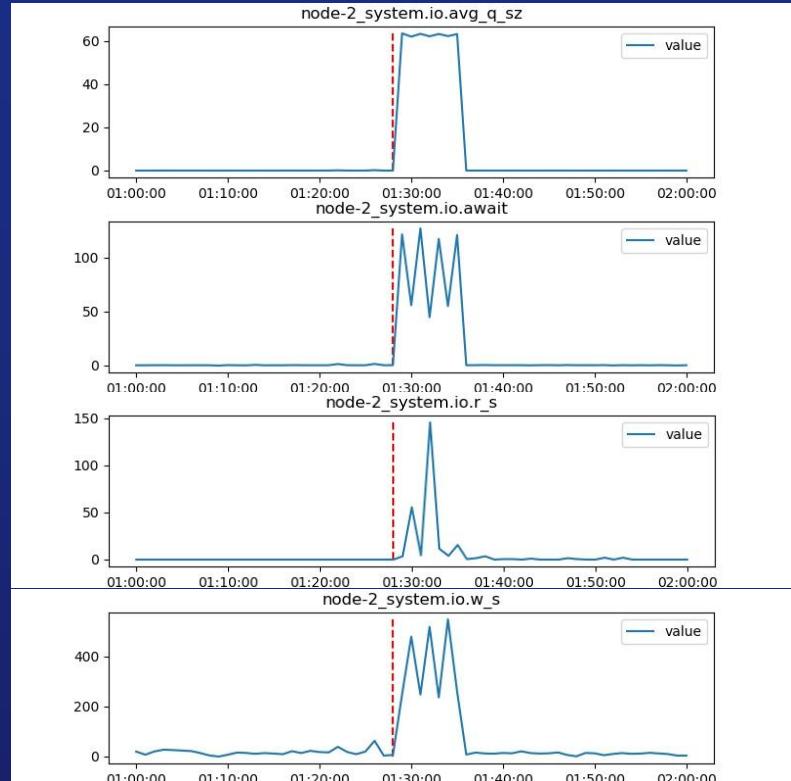
- 例如，2022/05/03 01:28:00, node-2, node 磁盘空间消耗



CPU相关指标



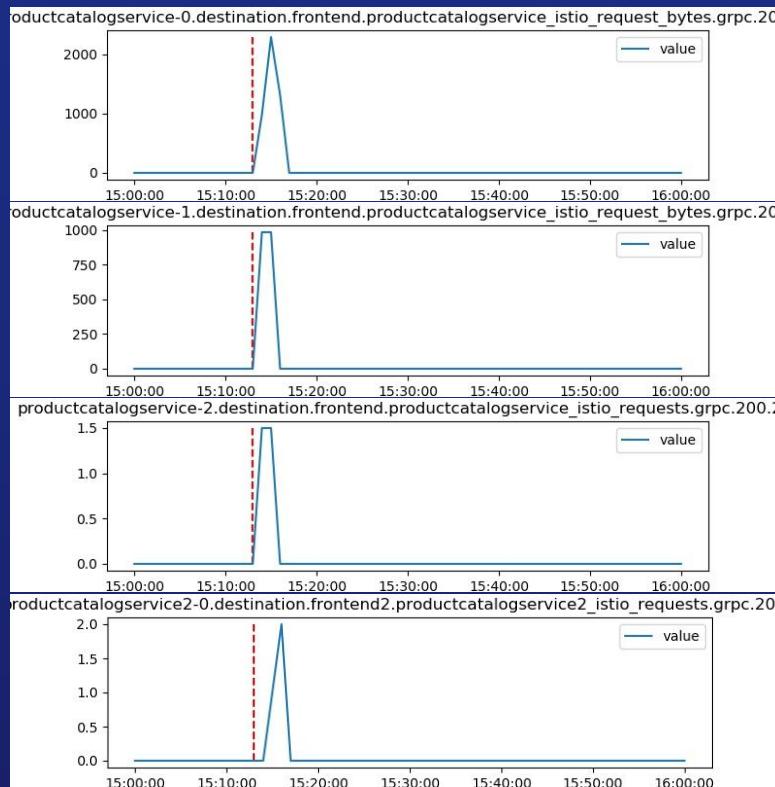
负载相关指标



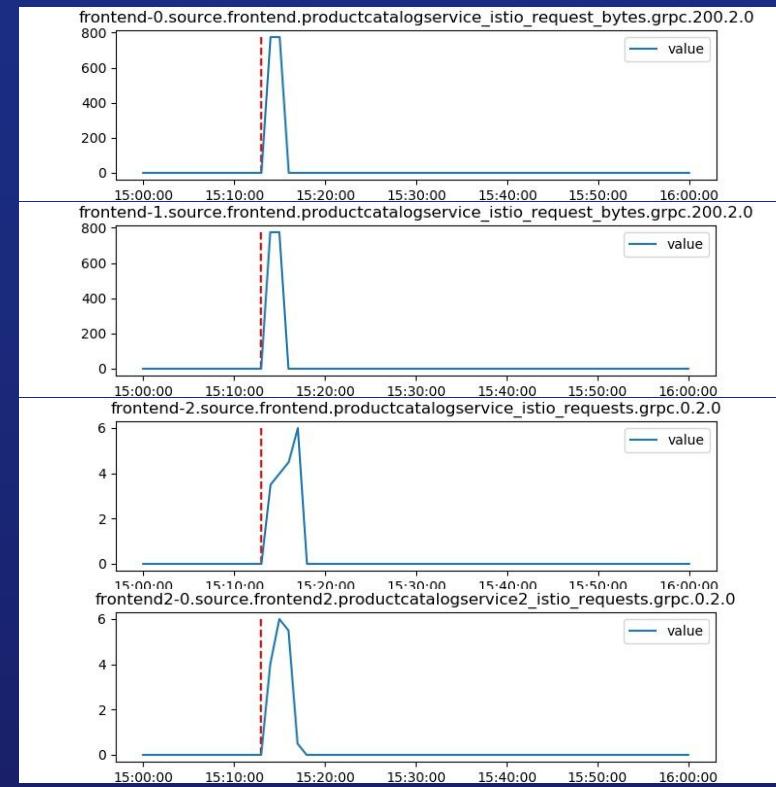
磁盘总IO、读写IO相关指标

## □ 赛题挑战三：调用关系间的故障传播

- 1) **istio调用关系故障传播**。例如，2022/05/03 15:13:00, productcatalogservice, k8s容器网络丢包。

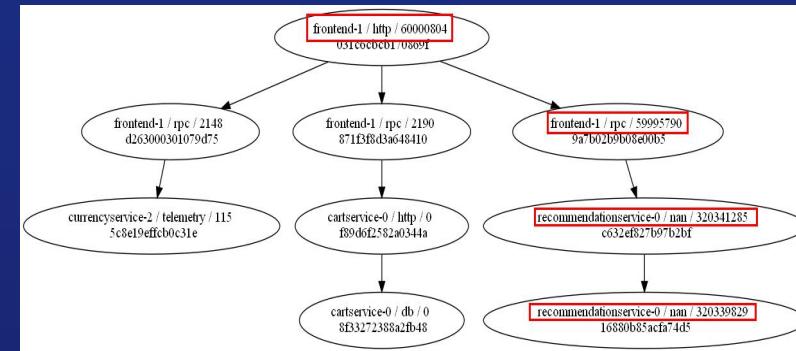


发生在productcatalogservice服务4个容器下的异常  
(frontend服务调用productcatalogservice服务)



发生在frontend服务4个容器下的异常 (frontend服务  
调用productcatalogservice服务)

- 2) **调用链调用时延故障传播**。例如，2022/03/24 01:03:00, recommendationservice-0, k8s网络资源包损坏。





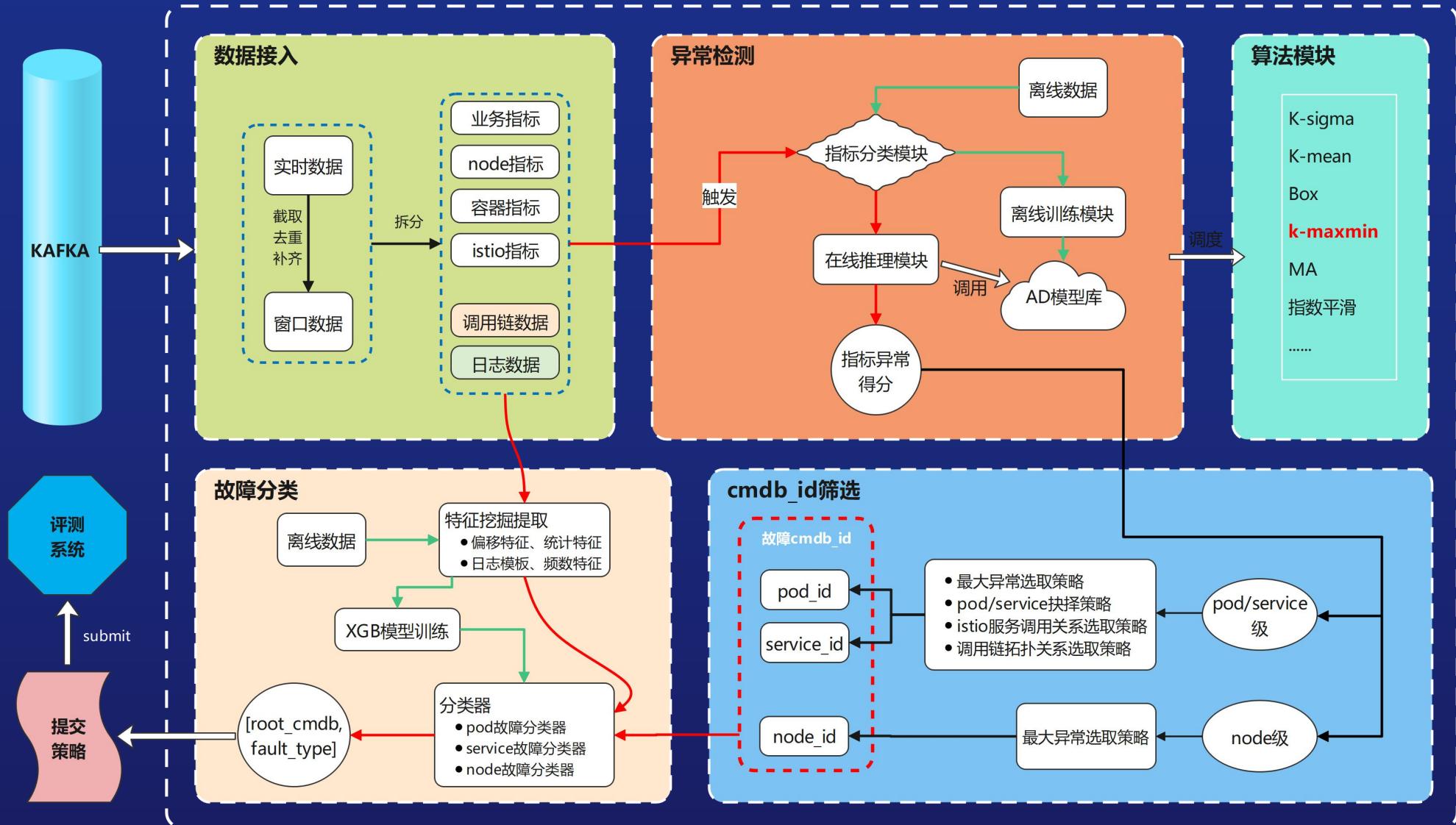
AIOps  
Challenge

2022 CCF国际AIOps挑战赛决赛  
暨AIOps研讨会

# 第三章节

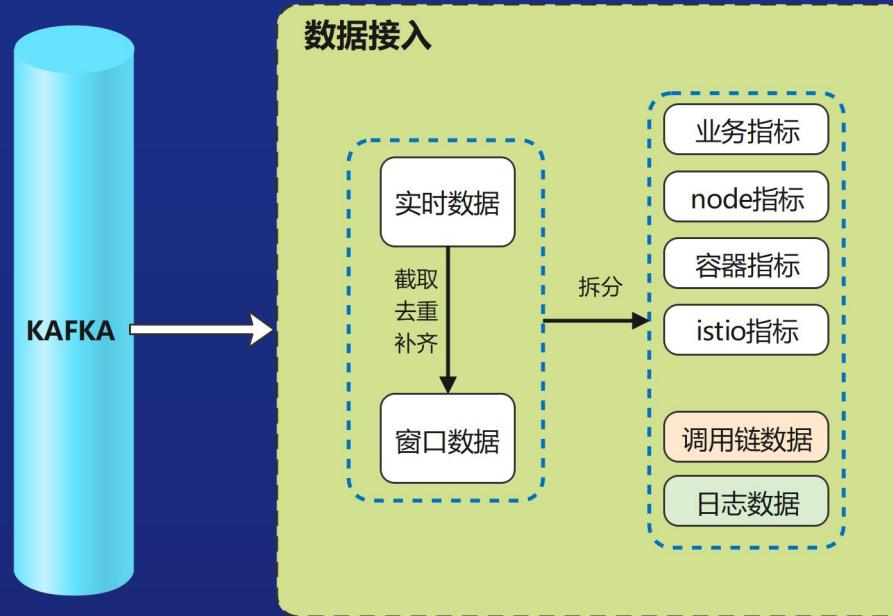
# 解决之道

# 总体方案



## □ kafka流，4个topic数据

- 黄金业务指标
- 性能指标
- 调用链
- 日志



## □ 数据持久化

- 黄金业务指标、调用链、日志 均缓存20分钟时间窗口数据，性能指标缓存65分钟数据。
- 对所有数据进行数据类型纠正、去重、时间线补齐。
- 对性能数据进行拆分：node性能数据、container性能数据、istio服务调用数据

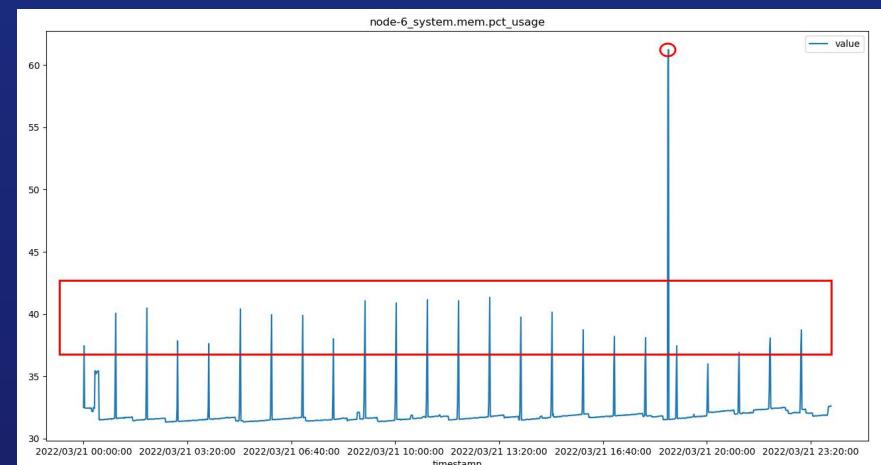
## □ 指标错综复杂，异常千奇百怪，何解？

- 分而治之
  - 对时序进行分类，不同的指标类型采用不同的算法进行异常检测



- 自研算法 **K-MaxMin**

- K-MaxMin异常检测算法是我们基于3Sigma原则改进的一种异常检测算法，用于不符合高斯分布的数据异常检测。例如在下图所示的场景中，数据不符合高斯分布。绝大部分数据近似两点分布，但是有极少数小突刺。如果直接使用箱体法或者3sigma原则则会将所有小幅度突刺全部作为异常进行误报。这种情况下使用K-MaxMin异常检测则能够精准检测出真正的异常点。



# CMDB\_ID选取策略

## □ 最大异常选取策略

- 顾名思义，即根据每个cmdb\_id所包含的所有指标异常程度来进行选取。
- 例如，2022/05/05 07:06:00，emailservice-2 k8s容器读io负载，检出如下异常分布：

```

container pod 2022/05/05 02:16:00 node-5.emailservice-2 29 {'node-5.cartservice-0': 4, 'node-5.checkoutservice-0': 1, 'node-5.emailservice-1': 3, 'node-5.emailservice-2': 29, 'node-5.paymentservice-0': 2, 'node-5.redis-cart2-0': 1, 'node-5.shippingservice-0': 1, 'node-5.shippingservice-1': 1, 'node-5.shippingservice-2': 2, 'node-6.adservice2-0': 1, 'node-6.currencyservice2-0': 1}
kpi node error num: 18
s_result : None c_result : node-5.emailservice-2 n_result : None
emailservice-2 ('k8s容器读io负载', 0.94553846) ('k8s容器内存负载', 0.35683802)
2022/05/05 02:15:31 cost time : 1.8578641414642334 submit root rca: ['emailservice-2', 'k8s容器读io负载']
{"code":0,"msg":"","data":1}

```

## □ pod/service抉择选取策略

- 这个策略主要是要区分当故障发生时，到底是pod故障，还是service故障。从service故障的定义得知：当某个service的4个pod都发生故障时，则认定为service故障。而实际情况可能某个service服务的3个或者2个pod发生严重故障，其service服务才是根因故障。
- 例如，2022/03/24 18:27:00，emailservice k8s容器网络资源包损坏，检出如下异常分布：

```

container service <--pod 2022/03/24 18:27:00 emailservice 36 {'node-3.currencyservice2-0': 1, 'node-3.paymentsservice-2': 1, 'node-3.shippingservice-2': 2, 'node-3.shippingservice2-0': 2, 'node-5.checkoutservice-0': 4, 'node-5.checkoutservice-1': 8, 'node-5.checkoutservice2-0': 2, 'node-5.currencyservice-1': 2, 'node-5.currencysevice-2': 2, 'node-5.emailservice-0': 20, 'node-5.emailservice-1': 4, 'node-5.emailservice-2': 12, 'node-5.paymentservice-0': 1}
('k8s容器网络资源包损坏', 0.99991655)
2022/03/24 18:27:00 submit root rca: ['emailservice', 'k8s容器网络资源包损坏']

```

# CMDB\_ID选取策略

## □ istio服务调用关系选取策略

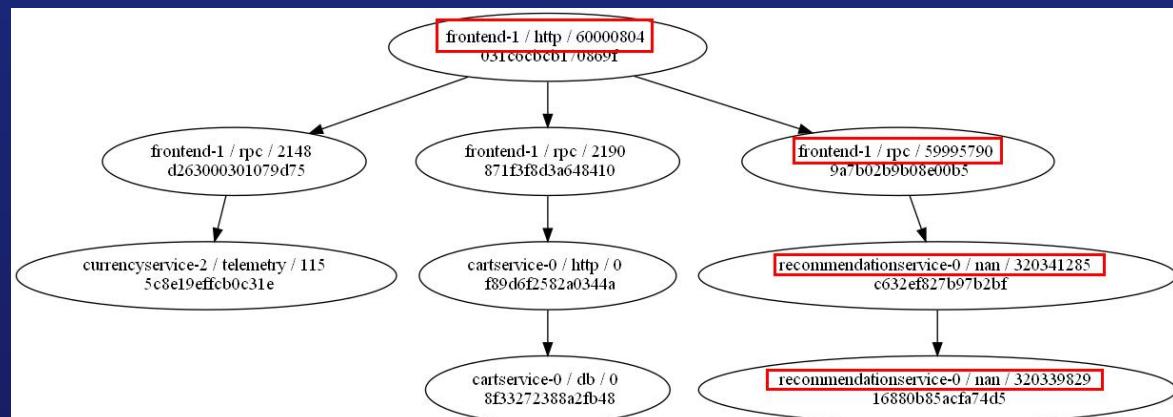
- istio的流量相关指标，cmdb\_id格式：**容器名.source/destination.服务调用方.被调用服务**。我们通过对istio数据异常检测，可同时获得3个异常得分子典，分别是**容器异常得分子典、服务调用方异常得分子典、被调用服务得分子典**。首先根据**服务调用方异常得分子典、被调用服务得分子典**确定是服务调用方异常还是被调用服务异常，然后结合**容器异常得分子典**确定是服务异常还是服务下的某个pod异常。
- 例如，2022/03/24 00:39:00，currencyervice-2 k8s容器cpu负载，检出如下异常分布：

```

2022/03/24 00:40:00 destination====p currencyervice-2 12
service destination ad dict: {'currencyervice': 10}
service source ad dict: {'checkoutservice': 4, 'frontend': 6, 'recommendationservice': 2}
pod ad dict: {'checkoutservice-2': 2, 'currencyervice-2': 4, 'frontend-0': 2, 'frontend-1': 2, 'recommendationservice-0': 2}
contain_istio: 2022/03/24 00:40:00 ['currencyervice-2', 'k8s容器cpu负载'] ('k8s容器cpu负载', 0.9988771)
2022/03/24 00:40:00 submit root rca: ['currencyervice-2', 'k8s容器cpu负载']
  
```

## □ 调用链拓扑关系选取策略

- 这个策略主要是要区分当故障发生时，被检测出来的cmdb\_id并不是根因，需要通过拓扑关系图来溯源。
- 例如，2022/03/24 01:03:00，recommendationservice-0 k8s网络资源包损坏。通过异常检测得分筛选出来的cmdb\_id为frontend-1。而调用链拓扑如右图：



# 故障分类

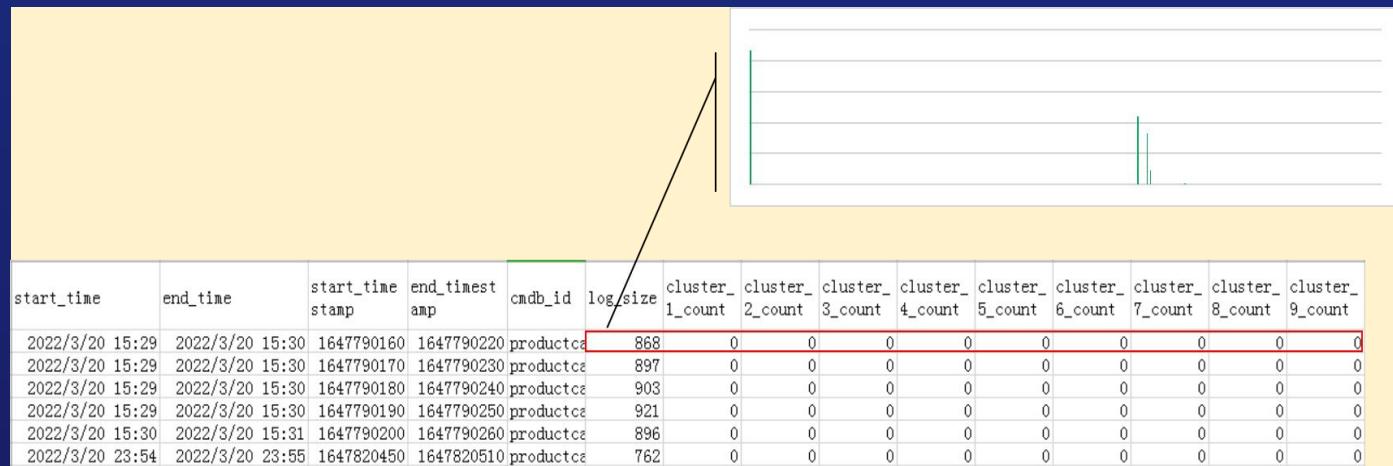
## 口 性能指标特征挖掘

- **pod/service级故障:** container特征（64个指标，64维）、container统计特征（3维）、node特征（59维）、node统计特征（3维），共129维。
- **node级故障:** 对于node，特征聚合方法与container相同，但没有container数据，只有node数据。包括node特征（59维），node统计特征（3维），共62维。



## 口 日志数据特征挖掘

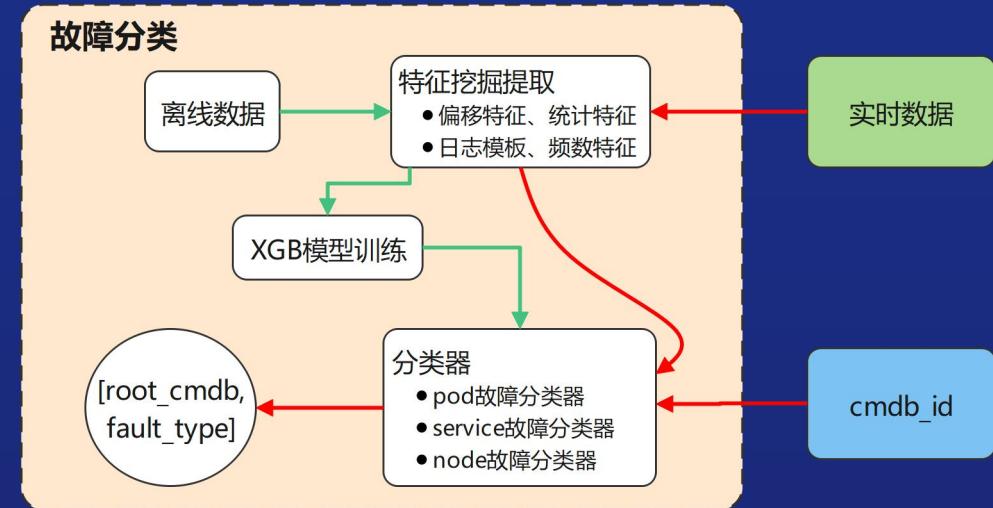
- Drain3挖掘所有日志模板并保存（共有n种日志模板）。
- 统计所有日志模板在窗口中出现的次数，作为频次特征。
- 另外补充统计特征：日志总数



start_time	end_time	start_time_stamp	end_time_stamp	cmdb_id	log_size	cluster_1_count	cluster_2_count	cluster_3_count	cluster_4_count	cluster_5_count	cluster_6_count	cluster_7_count	cluster_8_count	cluster_9_count
2022/3/20 15:29	2022/3/20 15:30	1647790160	1647790220	productca	868	0	0	0	0	0	0	0	0	0
2022/3/20 15:29	2022/3/20 15:30	1647790170	1647790230	productca	897	0	0	0	0	0	0	0	0	0
2022/3/20 15:29	2022/3/20 15:30	1647790180	1647790240	productca	903	0	0	0	0	0	0	0	0	0
2022/3/20 15:29	2022/3/20 15:30	1647790190	1647790250	productca	921	0	0	0	0	0	0	0	0	0
2022/3/20 15:30	2022/3/20 15:31	1647790200	1647790260	productca	896	0	0	0	0	0	0	0	0	0
2022/3/20 23:54	2022/3/20 23:55	1647820450	1647820510	productca	762	0	0	0	0	0	0	0	0	0

## □ 分类器

- 采用xgboost算法进行模型训练和推理。
- 形成三种分类器：**pod分类器**、**service分类器**、**node故障分类器**。
- 由于异常分类模块仅需处理异常检测输出的异常cmdb\_id，因此不需要处理正常数据，不需要处理数据不平衡问题，使得异常分类器的精度和召回率都可以达到较高的水平。
- 在cmdb\_id选取正确的情况下，线下F1得分高达**90%**，线上F1得分大概在**80%**左右。





AIOps  
Challenge

2022 CCF国际AIOps挑战赛决赛  
暨AIOps研讨会

# 第四章节

# 总结展望

# 总结

## 口 比赛情况

- 初赛TOP12
- 复赛TOP5
- 复赛提交次数：390次，前10名中，**提交次数最少**

① 概览		数据		排行榜
排名	团队名称	成员	分数	提交次数
1	Hi战队	zhangjk,LiuTa0,窗外的星星,GongYuhang,dfwv,jojo,188050564416,xuguangyao,便次工程师,code17	2411.75	436
2	浦智运维团队	wt19536,hjt0999,not+today,minduo111,yoeatfish602	1984.75	436
3	翼起飞	liukuan73,成胜,xingh,xiangda,wuqian	1954.75	513
4	中南-天云	csu_dhy,ty-liuxun,jianli123,chenzhao,yangping,hg00,amire,tclou,ud-jiangy,jiawehuang	1894.5	392
5	JustDo	hejiancsu,same_go,伊斯,xyx754,staygold,头上有天,alpha_bear	1887.75	390
6	AeroSpaceX	shayzego,chendh01,xiezhipeng,ayin1551,hubin666,Lawliet,看那是什么	1792.25	434
7	pa_tech_22	liwentao,xuanshou001,wubowen,aforwardzyu,zonglibo	1727.25	572
8	zsc8683	Zsc	1726.25	413
9	ABC_AIOPS	abc_zhang,donaldxuwm,木木,zhen,tree,syltaka,fuxiaopeng,gjbit	1713.25	494
10	benjili20224109	benjili2022,kingdompeak,linjiayu	1639	490



## □ 创新点总结

- **创新点一：**针对指标数据的多样性及复杂性，提出了一种多模态流式异常检测解决方案，采用分而治之的思想完成复杂时序的异常检测，用于快速发现故障。
- **创新点二：**自研设计 k-MaxMin 异常检测算法，很好的解决了伪周期类指标的异常检测误判问题，这是现有算法无法做到的。
- **创新点三：**针对K8S系统部署架构存在多层级（node/pod/service三个层级）对象，且各层级对象相互依赖等问题，提出了一种全方位cmdb\_id选取策略。包含如下4个子策略：
  - ① 最大异常选取策略
  - ② pod/service抉择策略
  - ③ istio服务调用关系选取策略
  - ④ 调用链拓扑关系选取策略
- **创新点四：**提出了多指标多数据特征挖掘方案（涉及偏移特征、统计特征、日志模块、频次特征等），并提出基于xgboost的高精度故障分类算法。线下F1得分高达90%，线上F1得分大概在80%左右。

## □ 改进思路

- **日志数据:**
  - 使用NLP分析日志value，用以解决未知模板的问题。
- **调用链数据**
  - 某些异常在指标上不明显，但在调用返回时长或调用次数上有特征。改进后，也可以作为故障发现的入口。
- **性能指标数据**
  - 通过简单高效的异常检测算法完成粗过滤，解决异常数据不均衡问题。
  - 结合少量标签，设计有监督异常检测算法。

## □ 落地应用

- IT领域自运维
- CT领域网络运维（智能根因定位、跨域定界定位等）



2022 CCF国际AIOps挑战赛决赛暨AIOps研讨会

# THANKS

