AIOps挑战赛答辩

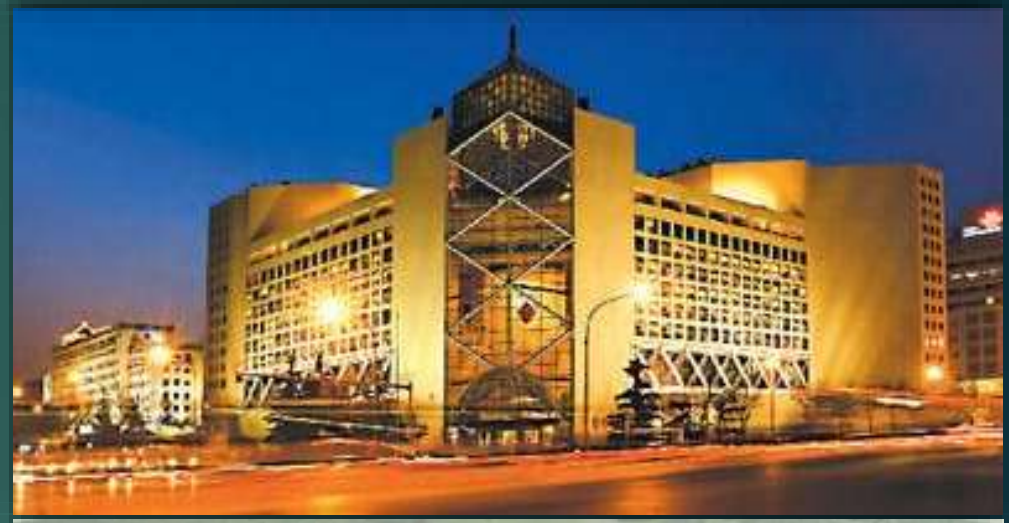
之 “微服务应用系统故障发现和根因定位”

掘金人战队 王林

2020年8月18日

队伍介绍

- 中国银行·信息科技运营中心·系统管理一团队
- 队员：王林、王鹏
- 队伍名称：掘金人
- 初赛排名：7
  复赛排名：3



赛题分析

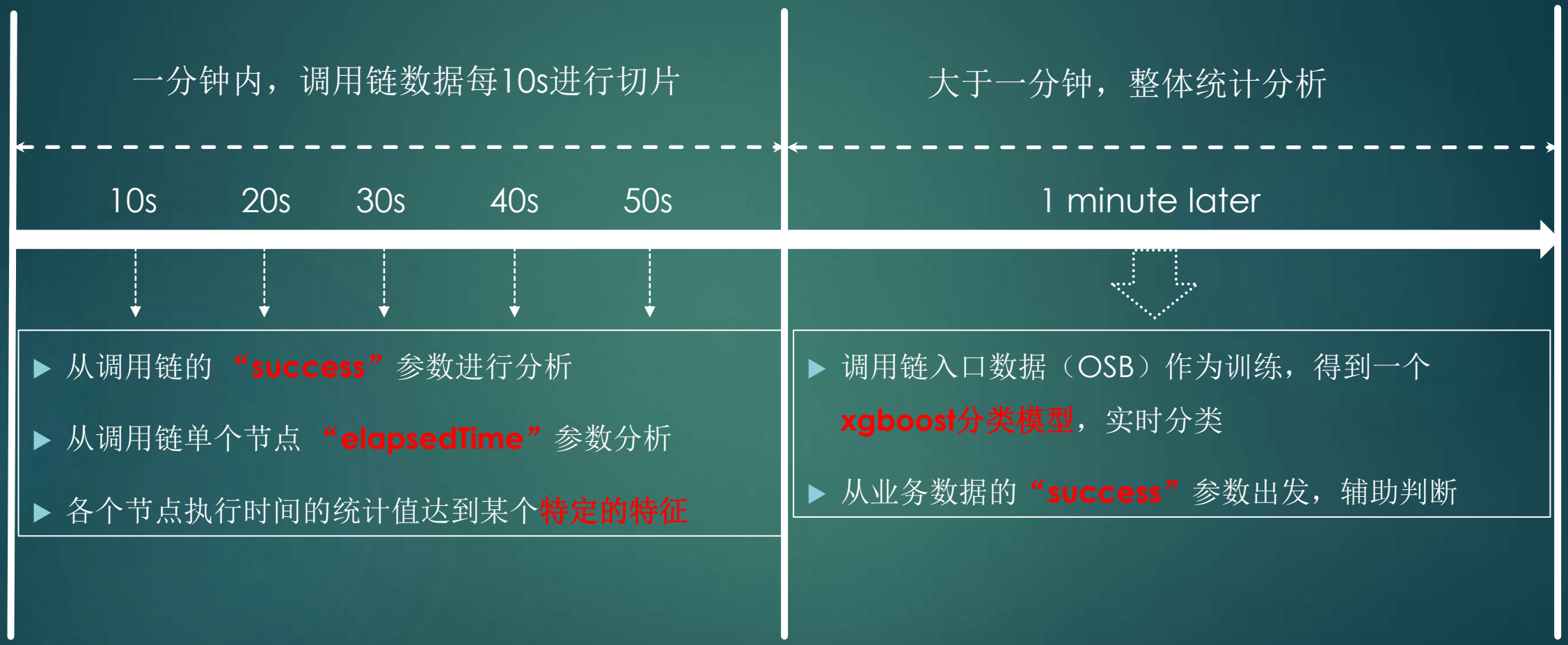
题目：微服务应用系统故障发现和根因定位

- 故障发现，即为故障检测
  - 利用业务数据，调用链关系数据，以及平台指标数据，对故障进行实时监测，及时发现故障
- 根因定位，即为故障定位
  - 利用调用链关系数据以及平台指标，在故障发生后进行故障定位，确定故障的类型、具体的节点以及平台指标

故障检测-难点

- **实时**的流式数据
- 故障持续时间不定，有可能**小于一分钟**
- 业务数据为对过去一分钟的描述，有一定的**延迟**

故障检测-解决方法


一分钟内，调用链数据每10s进行切片
大于一分钟，整体统计分析
10s
20s
30s
40s
50s
1 minute later
从调用链的 “success” 参数进行分析
从调用链单个节点 “elapsedTime” 参数分析
各个节点执行时间的统计值达到某个特定的特征
调用链入口数据（OSB）作为训练，得到一个xgboost分类模型，实时分类
从业务数据的 “success” 参数出发，辅助判断

故障检测-重点

- 调用链‘elapsedTime’的baseline
  - 对正常数据进行统计，找到每个调用链节点中的**P99**（第99百分位）作为baseline
  - 超过阈值后其对应节点的统计量会有变化

- xgboost模型
  - 特征包含一分钟内调用链OSB持续的**最大值**与**最小值**，**调用链数量**与**成功调用的数量**，**执行时间**，**每秒完成调用次数**，**平均时间**，**成功率**

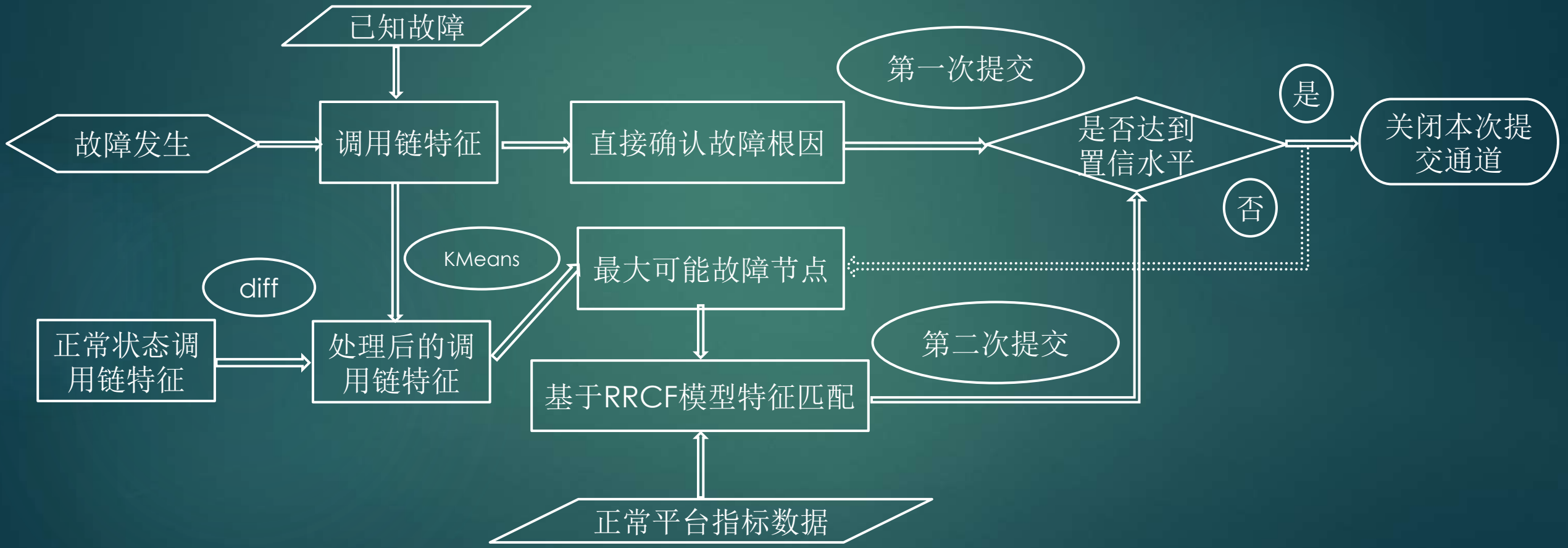
startTime	min	max	num	succee_num	elapsed_time	count_per_sec	avg_time	succee_rate
7/16/2020 22:00	0	46	632	632	414288	13.73913043	0.7115	1
7/16/2020 22:01	0	24	343	343	230467	14.29166667	0.622	1
7/16/2020 22:02	0	24	359	359	230579	14.95833333	0.6473	1
7/16/2020 22:03	0	25	359	359	227727	14.36	0.6253	1
7/16/2020 22:04	0	23	385	385	223830	16.73913043	0.6668	1
7/16/2020 22:05	0	24	399	399	229224	16.625	0.6425	1

```
params = {'learning_rate': 0.2,
          'n_estimators': 75,
          'max_depth': 1,
          'min_child_weight': 0.1,
          'seed': 0,
          'subsample': 1.0,
          'colsample_bytree': 0.4,
          'gamma': 0,
          'reg_alpha': 0.1,
          'reg_lambda': 0.04,
          'scale_pos_weight': 2}
```

故障定位-难点

- Interval大都在**60s以上**，与检测到的故障时间点无法完全吻合，有一定的延迟
- 平台指标数据变化的幅度大小与其是否是根因**并不完全相关**
- 网络故障往往同时体现在**多个平台指标**上

故障定位-解决方法


已知故障
故障发生
调用链特征
直接确认故障根因
第一次提交
是否达到
置信水平
是
关闭本次提
交通道
否
KMeans
最大可能故障节点
diff
正常状态调
用链特征
处理后的调
用链特征
第二次提交
基于RRCF模型特征匹配
正常平台指标数据

故障定位-重点

- 调用链处理
  - 基于baseline，得到某一个时间段内的所有节点的一个统计特征分布，**应用K=1的KNN算法**，得到匹配的特征，进而得到其对应的类别（故障根因）
  - 记录正常情况下的调用链特征，与异常情况下的调用链特征做diff，之后通过**K-Means聚类**，得到最大可能故障节点，作为RRCF模型的输入
- 基于**RRCF模型**的特征匹配
  - 故障发生时，对输入节点的所有平台指标分别通过RRCF得到某个分值，合并得到组合特征，后匹配具体的特征（**网络特征，CPU特征，内存特征等**），得到故障类型，输出根因指标。

举例说明

start_time	sec	os_021	os_022	db_003	db_007	db_009	docker_001	docker_002	docker_003	docker_004	docker_005	docker_006	docker_007	docker_008
7/16/2020 22:56:00	0	0	10	128	11	96	45	165	10	0	0	142	0	0
7/16/2020 22:56:10	10	0	0	107	3	56	54	86	9	19	2	115	0	0
7/16/2020 22:56:20	20	0	0	157	22	111	24	212	36	25	0	187	0	0
7/16/2020 22:56:30	30	0	0	118	85	121	29	286	0	0	0	140	0	0
7/16/2020 22:56:40	40	0	5	76	36	112	21	366	16	8	0	88	0	0
7/16/2020 22:56:50	50	0	0	120	44	196	27	330	29	0	2	144	0	0

db_003、db_007、db_009、docker_002、docker_006升高，特征匹配到**os_018 Sent_queue**

```
[['os_018', 'Sent_queue']]
submit for 1 time(s)
```

1分钟内，第一次提交，但此时置信水平低

```
pred 0.99071056
judge false traces ['os_018']
rrcf: [['os_018', 'Sent_queue']]
rrcf total time is : 0:00:06.699891
```

```
[['os_018', 'Sent_queue']]
same answer,submit ends
```

1分钟后，第二次尝试提交，与第一次结果相同，关闭提交通道

工程实现

- 基于python语言，2500+行代码
- 利用主办方提供的consumer.py进行kafka数据的接收
- 所有的逻辑处理都在子线程上
  - 不影响主线程消息接收
  - 出现问题不影响主线程继续运行
  - 加锁，避免多个子线程访问同一数据集的情况
- 效率至上，所有的数据都不落地，都保存在内存当中
- 充分利用现有资源，在需要并行计算的时候实现负载均衡
- 内存回收机制

谢谢

Q&A