



2021 国际AIOps挑战赛决赛  
暨AIOps创新高峰论坛

# 鲁棒时序异常检测及智能根因定位

铃动时序智能团队

阿里巴巴达摩院&云数据库



第一届国际互联网产业科技创新大会暨互联网创新产品展览会  
The First International Internet Industry Science And Technology Innovation Conference & Internet Innovation Product Exhibition



=

達摩院

ALIBABA DAMO ACADEMY

+



阿里云

铃动(Lindorm)时序智能

时间序列分析算法

Lindorm时序数据库

## 团队成员

何凯（领队）、周城、文波

达摩院决策智能实验室 & 阿里云数据库NoSQL产品部

## 特别鸣谢

浙江大学 陈岭教授团队 (陈东辉、郑汉邦、崔家华)

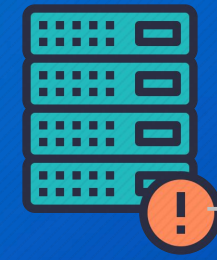
大连理工大学 刘胜蓝教授团队 (张爱宾、李运恒、刘铠源)

## 比赛成绩


复赛成绩第3名，其中B系统得分排第1名



待检测数据



性能指标: CPU/Mem等



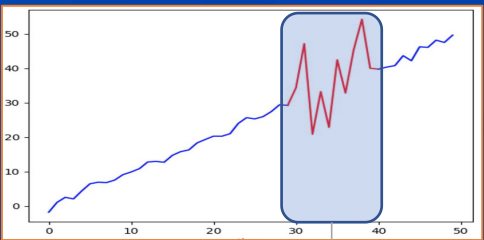
服务日志

```
10.1.168.193 -- [01/Mar/2012:15:52:07 +0800]
"GET /Send?AccessKeyId=8225105404 HTTP/1.1" 200
0.04 "-" "Mozilla/5.0 (X11; Linux i686 on x86_64;
rv:10.0.2) Gecko/20100101 Firefox/10.0.2"

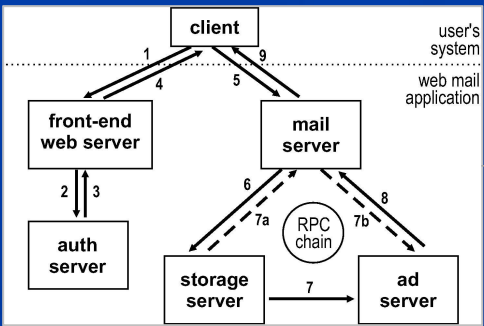
10.1.168.193 -- [01/Mar/2012:15:52:11 +0800]
"GET /Send?AccessKeyId=8225105404 HTTP/1.1" 500
0.08 "-" "Mozilla/5.0 (X11; Linux i686 on x86_64;
rv:10.0.2) Gecko/20100101 Firefox/10.0.2"
```

根因指标  
(直接定位)

KPI/黄金指标

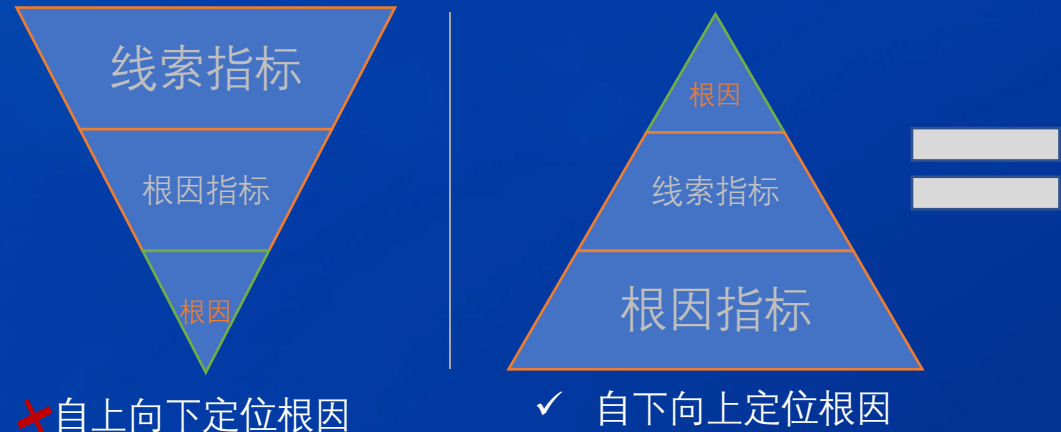


线索指标  
(间接定位)



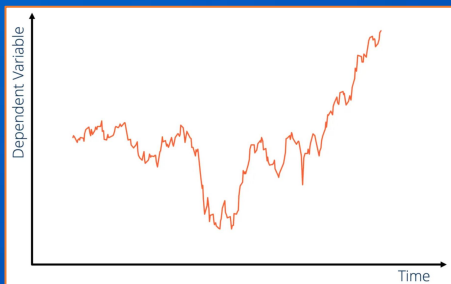
调用链Trace

从 赛题/业务 要求出发

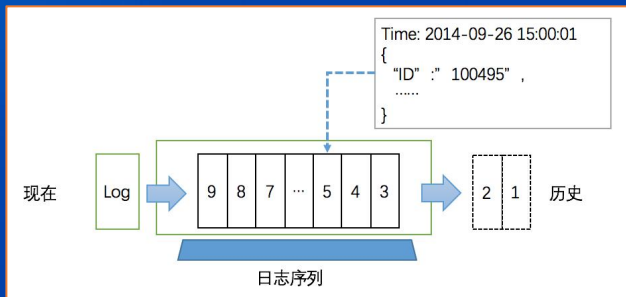


# 一切皆为时间序列

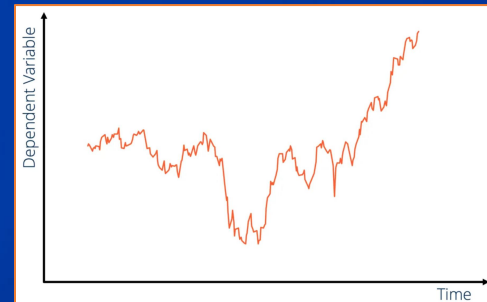
## ➤ 时间序列数据源



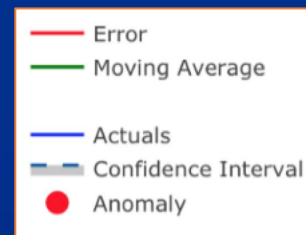
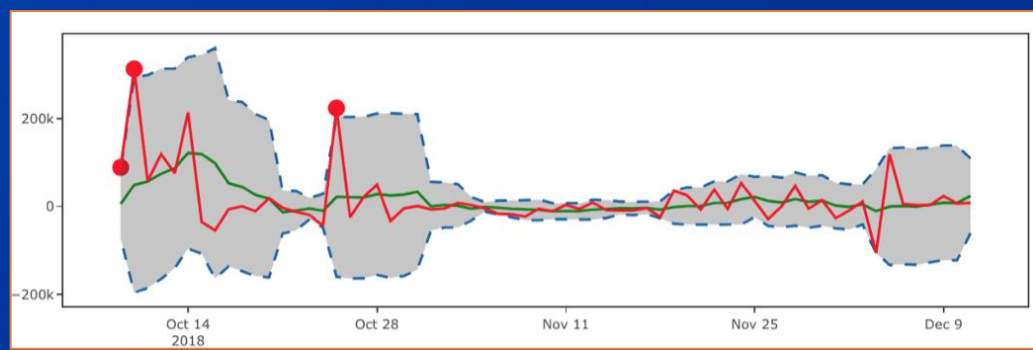
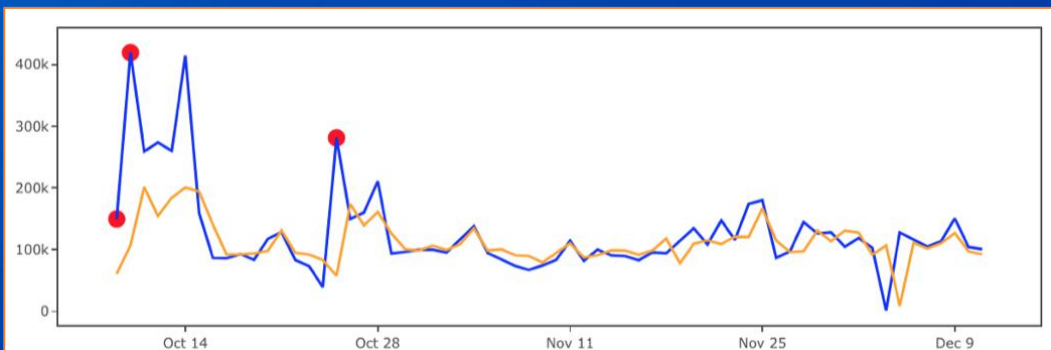
✓ KPI/性能指标：直接时序数据



➤ 日志/调用链：间接时序数据



## ➤ 基于时间序列的异常检测



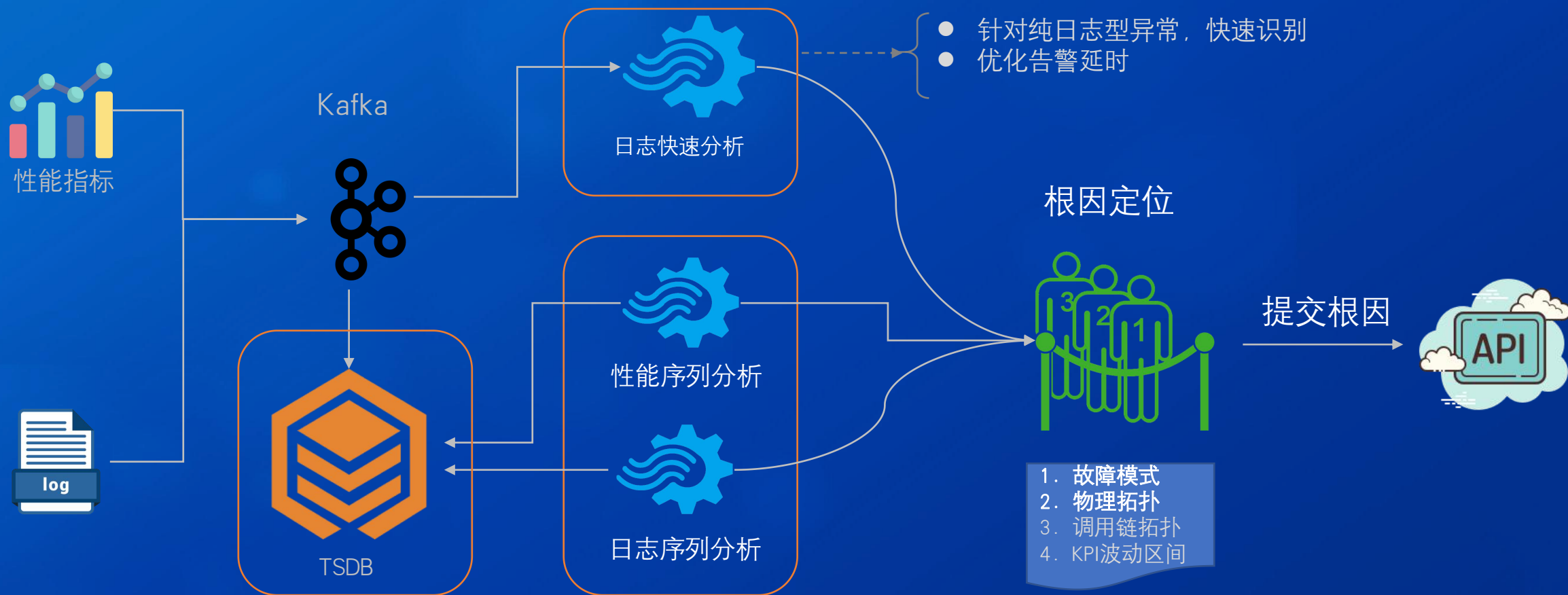
## ➤ 难点与挑战

- 高效，鲁棒性的时序异常检测算法
- 如何基于业务知识从间接数据中提取时间序列

- 如何自动识别每种故障类型所涉及的相关时间序列
- 针对每种故障类型，如何自动调参，提升精度和召回
- 如何定位根因(单个故障会导致多个序列/cmdbid产生异动)



# 计算架构 - 整体方案



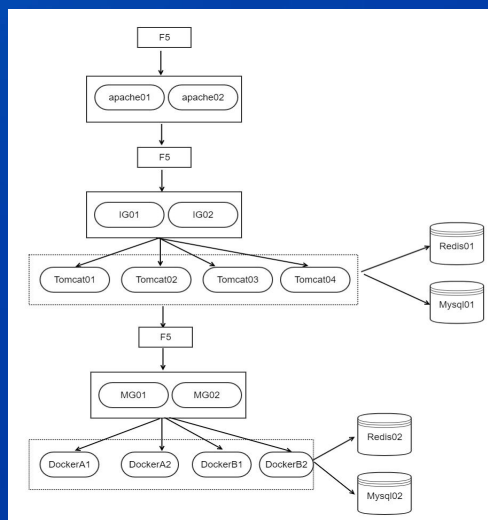
- ETL处理: 提取合规时间序列
- 窗口处理: 削峰填谷, 抹平乱序
- TTL: 数据超时清理

- 并行分析: 无复杂数据交互, 按机器(cmdb\_id) + 数据类型 处理
- 故障识别与分类: 支持嵌入运维人员知识
- 简单易扩展: 通过 \*离线学习\* 训练参数后即可支持新系统(数据)

当一个异常窗口内，同时检测到多个类型的故障时，需要合理定位根因

具体示例：

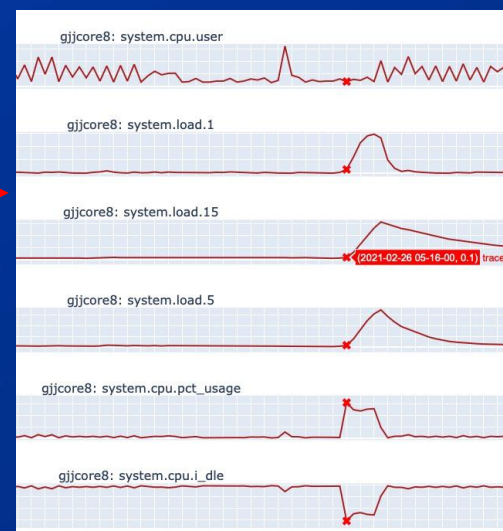
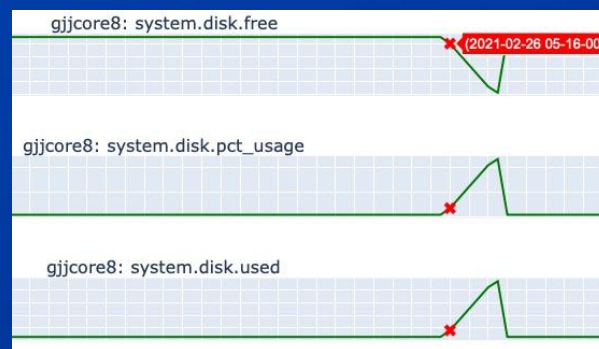
## ❑ 基于拓扑图上下游关系



0304 10:05系统B Mysql01网络隔离导致上游Tomcat同时异动，此时需要利用拓扑结构对该时段内的伪根因进行屏蔽

- ❑ 基于节点**拓扑关系**定位最可能根因
- ❑ 基于**故障模式**定位最可能根因
- ❑ 其他：调用链拓扑，KPI波动区间

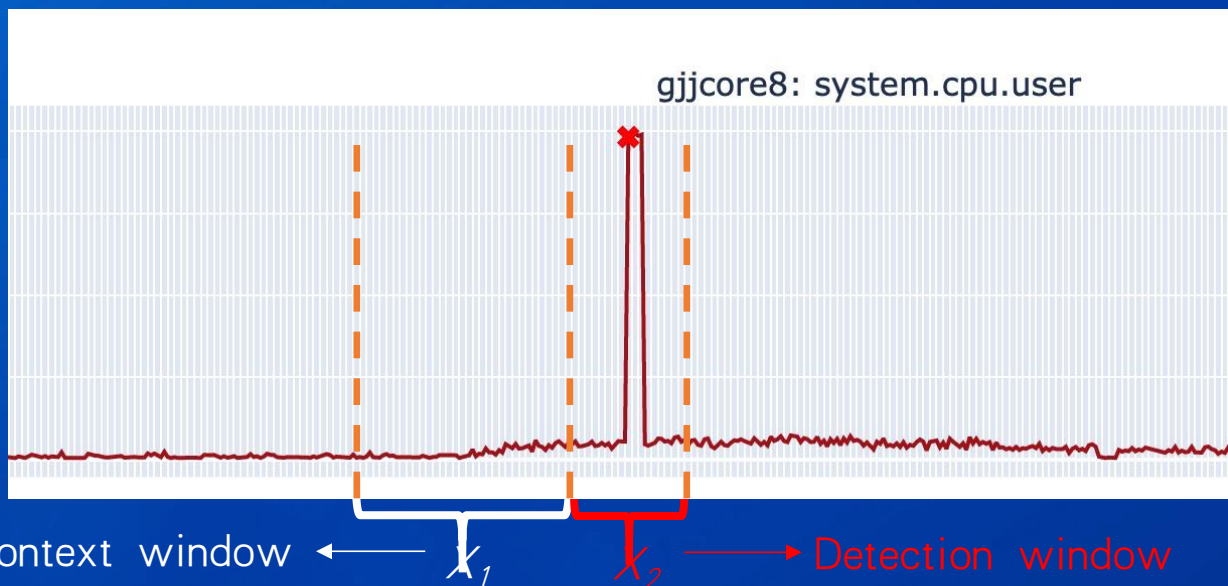
## ❑ 基于故障模式



0226 05:16系统A磁盘故障可以引起同时段内的CPU负载和使用率高，据此可以离线学习和利用专家知识提前为故障类型确定优先级，如磁盘>CPU，内存>CPU等

## 通用鲁棒时序异常检测算法框架

- 时序数据周期识别、趋势提取与残差分解（如STL算法等）
- 对残差项使用Robust T-test进行离群点检测
- 利用已知故障标签数据基于Grid Search进行自动参数调节



优点:

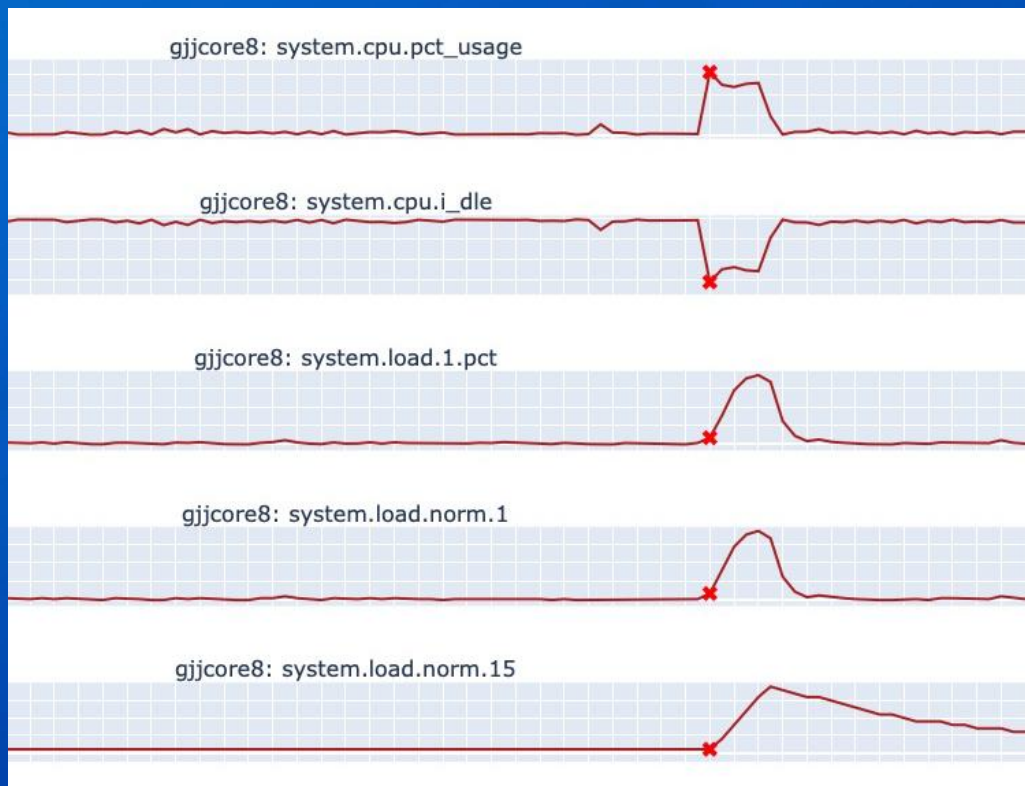
- ✓ 配合中位数滤波、双边滤波等可以进一步提高检测的鲁棒性
- ✓ 计算速度快，可以使用online版本增量更新检测
- ✓ 参数少，通用性强，减少自动调参的时间
  - ✓ 左右窗口长度
  - ✓ 统计显著性水平
  - ✓ 指标上升还是下降异常，如cpu使用率检测上升异常，内存闲置量检测下降异常

$$t = \frac{\bar{X}_1 - \bar{X}_2}{s_p \cdot \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

where

$$s_p = \sqrt{\frac{(n_1 - 1) s_{X_1}^2 + (n_2 - 1) s_{X_2}^2}{n_1 + n_2 - 2}}$$

公式中的 $X_1$ 和 $X_2$ 分别对应左右窗口中的鲁棒统计数值，在异常检测中有高易用性和有效性



❑ 基于FastDTW算法和皮尔逊相关系数定位相关的性能指标集合

```
Failures_candidates = []
```

```
For failure_type in ['cpu', 'disk', 'mem', 'net', 'JVM']:  
    if # of anomaly metrics w.r.t. failure_type > threshold:  
        Failures_candidates.append(failure_type)
```

离线学习到的与每种故障相关的指标集合

离线学习到的跟故障类型相关的阈值信息，只有超过一定数量的相关指标同时报警才会认为该类故障类型发生

优点：

- ✓ 进行故障定位时，只扫描跟每种故障类型相关的指标集合，极大缩短根因定位范围
- ✓ 只监测部分指标，就能够推断出其余相关指标的变化，从而减少指标检测数，降低计算开销





由于每种故障均会持续一段时间，实时在线检测场景中一种故障类型可能会出现重复报警的情况  
重复提交答案会占用提交次数，并且带来时延，降低得分，所以我们设置了如下的报警抑制函数

if 异常分2≤异常分1, then 不提交新报警

异常分 = 预估根因数/延时

该方法在运维领域也常用，能够极大的抑制重复的报警

## 报警抑制前

```
所有 cmdbid [      定位时间      , 延迟(s),报出根因,命中根因,原始根因]
gjjcore8 [('2021-02-26 05:17:00', 58, 3, 0, 3), ('2021-02-26 05:18:00', 118, 3, 3, 3), ('2021-02-26 05:20:00', 238, 3, 3, 3)]
gjjcore2 [('2021-02-26 05:59:00', 60, 3, 3, 6), ('2021-02-26 06:08:00', 600, 10, 0, 6)]
gjjha2 [('2021-02-26 07:21:00', 120, 11, 11, 13)]
gjjcore8 [('2021-02-26 10:48:00', 60, 10, 10, 10)]
gjjcore8 [('2021-02-26 13:09:00', 0, 9, 9, 12), ('2021-02-26 13:11:00', 120, 9, 9, 12)]
gjjcore9 [('2021-02-26 23:28:00', 60, 10, 9, 10), ('2021-02-26 23:37:00', 600, 3, 0, 10)]
```

## 报警抑制后

```
所有 cmdbid [      定位时间      , 延迟(s),报出根因,命中根因,原始根因]
gjjcore8 [('2021-02-26 05:17:00', 58, 3, 0, 3), ('2021-02-26 05:18:00', 118, 3, 3, 3)]
gjjcore2 [('2021-02-26 05:59:00', 60, 3, 3, 6)]
gjjha2 [('2021-02-26 07:21:00', 120, 11, 11, 13)]
gjjcore8 [('2021-02-26 10:48:00', 57, 10, 10, 10)]
gjjcore8 [('2021-02-26 13:09:00', 0, 9, 9, 12)]
gjjcore9 [('2021-02-26 23:28:00', 60, 10, 9, 10)]
平均根因F1 0.9392909356725146
```

## 技术亮点总结：

- ✓ 使用鲁棒性的时序异常检测算法，有效处理高噪音数据
- ✓ 算法通用性强，可以进行效果的快速复制和大规模部署，通过使用Robust T-test + RobustPeriod<sup>1</sup>周期判断/RobustSTL<sup>2, 3</sup>数据分解，已经在阿里内部成功落地多项场景
- ✓ 设计自动参数调节进一步提高通用性
- ✓ 自动挖掘指标间的相关性，提炼与故障最相关的指标集合
- ✓ 设计巧妙的Ranking函数，只在增益较大时才提交和覆盖先前答案

## 成绩总结：

- 复赛总成绩第3
- 复赛系统B得分排第1名

## 未来的提升方向：

- ❑ 结合深度因果关系与运维专家知识，发掘更多的故障模式。
- ❑ 对多指标系统进行整体建模和状态分析，如结合所有的数据源信息对系统当前的状态进行表征，再进行异常检测。

## 阿里达摩院决策智能实验室：

1. Qingsong Wen, Kai He, Liang Sun, Yingying Zhang, Min Ke, and Huan Xu, "RobustPeriod: Time-Frequency Mining for Robust Multiple Periodicities Detection," in Proc. ACM SIGMOD International Conference on Management of Data (SIGMOD 2021)
2. Qingsong Wen, Jingkun Gao, Xiaomin Song, Liang Sun, Huan Xu, Shenghuo Zhu. RobustSTL: A Robust Seasonal-Trend Decomposition Algorithm for Long Time Series. Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI 2019)
3. Qingsong Wen, Zhe Zhang, Yan Li and Liang Sun, "Fast RobustSTL: Efficient and Robust Seasonal-Trend Decomposition for Time Series with Complex Patterns," in Proc. of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2020), San Diego, CA, Aug. 2020.



2021 国际AIOps挑战赛决赛暨AIOps创新高峰论坛

# THANKS

谢谢观看



第一届国际互联网产业科技创新大会暨互联网创新产品展览会  
The First International Internet Industry Science And Technology Innovation Conference & Internet Innovation Product Exhibition

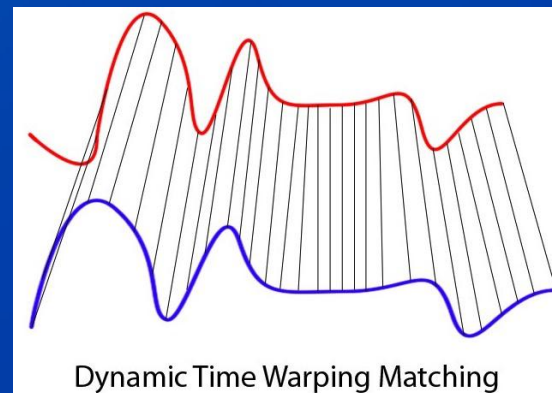


两个系统中的性能指标种类和数量繁多

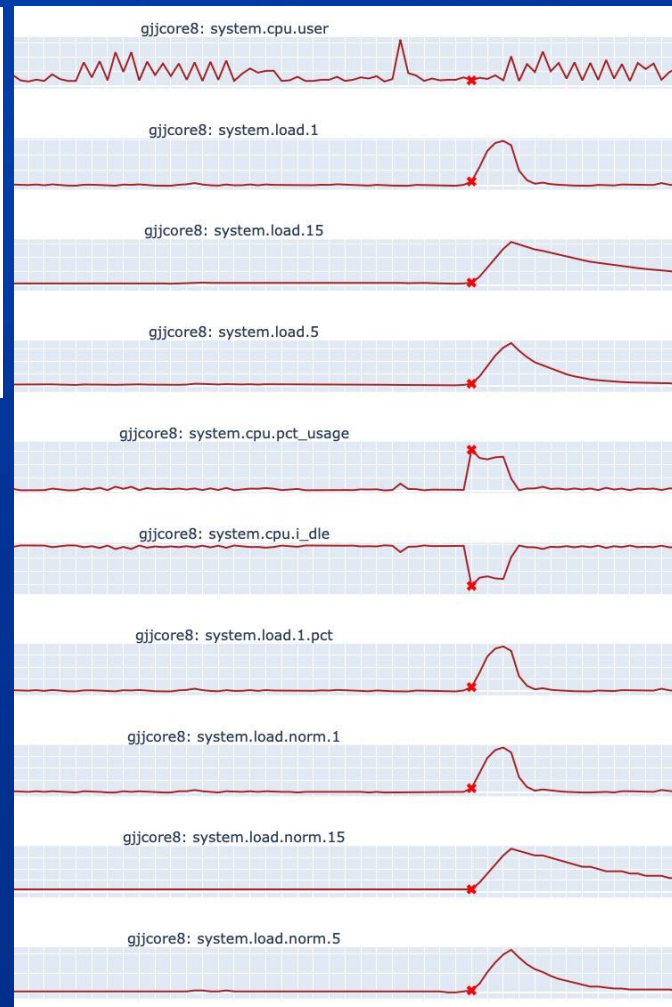
- ❑ 我们设计了基于DTW的算法自动挖掘与几种故障类型最相关的性能指标集合
- ❑ 针对集合中的每个指标，通过皮尔逊相关系数找寻与其最相关的其他指标
- ❑ 进行故障定位时，只扫描跟每种故障类型相关的指标集合，极大缩短根因定位范围

皮尔逊(Pearson)相关系数用来衡量两个变量间的线性关联程度。使用其可以只监测一部分指标，就能够推断出其余相关指标的变化，从而减少指标检测数，降低计算开销

$$r_{xy} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$



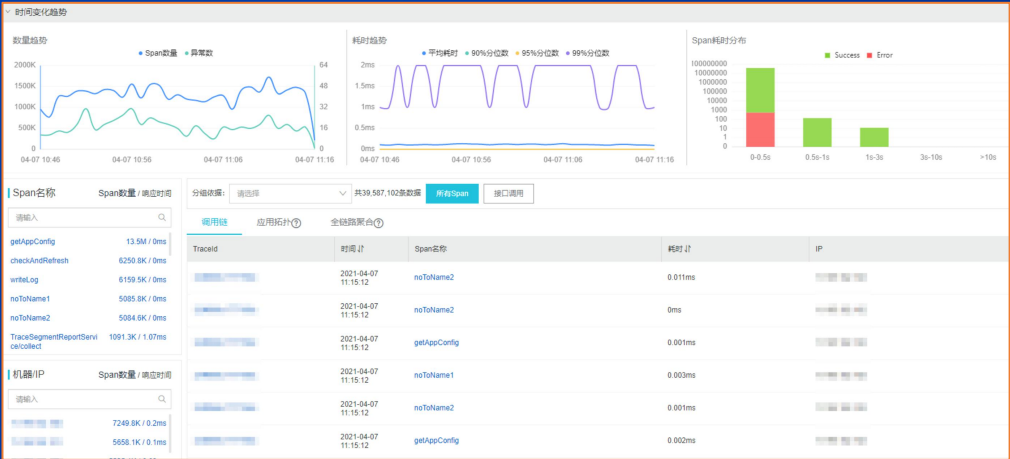
考虑到跟故障相关的指标可能在时间轴上存在时延，我们通过计算每对性能指标在故障时刻左右的DTW距离来定位与目标时间序列相关但是存在时延的指标集合



# FAQ: 黄金指标/调用链数据 是否可以充分利用？

- 黄金指标：提供 异常时间范围作为参考信息，可以在根因定位时用于过滤无关异常(噪音)
- 调用链数据：
  - ✓ 拓扑分析：还原节点调用关系，用于根因定位时提供拓扑信息(但是竞赛已经直接提供拓扑结构)。

✓ 水平分析：分析每条链中节点的调用耗时，标识机器ID(cmdb\_id)是否出现异常，可以在根因定位时用于过滤。



✓ 垂直分析：针对每个调用节点 (Span)进行聚合，转化为每个服务执行情况的时间序列后分析。

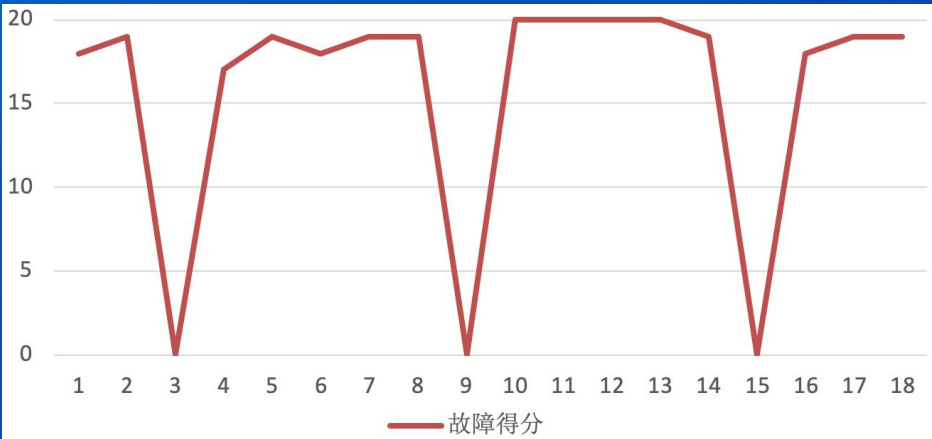
调用链

应用拓扑?

全链路聚合?

Span名称	应用名称	请求数/请求比例 ①	Span数量/请求倍数 ②	平均自身耗时/比例 ③	平均耗时	异常数/异常比例 ④
▼ GET		352 / 100.00%	352 / 1.00	0.58ms / 0.23%	241.069ms	0 / 0.00%
▼ LogManagerAction.doQueryView		67 / 19.03%	67 / 1.00	235.10ms / 18.56%	235.130ms	0 / 0.00%
Render.JsonValve.name		67 / 19.03%	67 / 1.00	0.03ms / 0.00%	0.034ms	0 / 0.00%
GET		200 / 56.81%	200 / 1.00	341.27ms / 80.43%	341.267ms	0 / 0.00%
Render.JsonValve.name		59 / 16.76%	59 / 1.00	0.03ms / 0.00%	0.028ms	0 / 0.00%
▼ LogManagerAction.doSearchByName		8 / 2.27%	8 / 1.00	80.48ms / 0.75%	80.520ms	0 / 0.00%
Render.JsonValve.name		8 / 2.27%	8 / 1.00	0.04ms / 0.00%	0.042ms	0 / 0.00%
▼ POST		234 / 100.00%	234 / 1.00	433.85ms / 14.28%	3036.939ms	0 / 0.00%
▼ LogManagerAction.doQueryRate		90 / 38.46%	90 / 1.00	2343.07ms / 29.67%	3408.988ms	0 / 0.00%
Render.JsonValve.name		83 / 35.47%	83 / 1.00	0.13ms / 0.00%	0.131ms	0 / 0.00%
▼ SourceSQLUtils.sql		90 / 38.46%	90 / 1.00	668.13ms / 8.46%	1065.796ms	0 / 0.00%
▼ GET		90 / 38.46%	90 / 1.00	0.02ms / 0.00%	397.669ms	0 / 0.00%
GET		90 / 38.46%	90 / 1.00	397.65ms / 5.03%	397.652ms	0 / 0.00%

- 故障识别率: 被捕获的故障模式, 准确性90%+, 命中得分Top 3



- 故障识别率不低的情况下, 总得分为何只是第三?
  - 系统A, 故障模式识别不够全面.
  - 系统A 与 系统B的故障注入数量不一样. 系统A多了1/3左右, 但总成绩只是两个系统相加.
  - 更多考虑通用性的异常识别与准确率, 对某些专有场景,缺少定制 的识别规则

复赛 排名	队名	通用性完成 情况
1	LR-AI0ps	基础要求
2	一行 bug	进阶要求
3	铃动时序智能	进阶要求