

Project PPT Notes

2020年11月30日 17:27

Tsinghua

- KDE for Anomalies Detection
- Topology for anomaly node location
- They added personalized rules to their models and its not told

H3C

- Locate which 网元 has problem. Example D1 calling D2 takes long. Then D3 call D2 takes long too. Then D2 has a problem. Similar for others
- Offline Preprocessing

BOCOIOPS

- Data -> Filtering -> Multi Threading -> Caching -> Root Cause Detection
- The call time that takes the longest
- Analysis Call steps relationship and success or not
- Checking on dockers, if dockers in same os has problem, then OS has problem

措施4：移动差分的故障检测定位方法

核心思想：

1、统计1分钟内，各种故障的调用链个数，在所有调用链的占比。

2、对同类型故障按时间进行差分计算

差分数据：1、基于临近点 2、基于移动均值

判断依据：一次单调上升，二次单调上升，然后根据不同阈值进行异常判断

| TYPE | REASON | PERCENT |
|--------|------------|---------|
| NET | docker_001 | 9.37 22 |
| CPU | docker_001 | 8.77 25 |
| CPU | docker_002 | 7.02 20 |
| CPU | docker_004 | 1.40 4 |
| CPU | docker_007 | 1.40 4 |
| NET | docker_002 | 0.88 2 |
| NET | docker_003 | 0.70 2 |
| RENOTE | docker_002 | 0.70 2 |
| CPU | docker_003 | 0.35 1 |
| CPU | docker_005 | 0.35 1 |
| CPU | docker_006 | 0.35 1 |
| CPU-DB | docker_007 | 0.35 1 |
| CPU-DB | docker_005 | 0.35 1 |
| RENOTE | docker_004 | 0.35 1 |
| CPU-DB | docker_006 | 0.18 0 |

| REASON | TYPE | TIME | PERCENT | NUM | TOTAL | DIFF | DIFF2 | DIFF3 | DIFF4 |
|------------|------|---------------------|---------|-------|-------|-------|-------|--------|-----------|
| docker_006 | NET | 2020-07-19 22:26:00 | 1.33 | 18.0 | 1349 | -0.07 | -0.05 | 0.000 | inf |
| docker_006 | NET | 2020-07-19 22:27:00 | 0.86 | 11.5 | 1302 | -0.37 | -0.28 | 1.363 | 0.700000 |
| docker_006 | NET | 2020-07-19 22:28:00 | 0.81 | 10.5 | 1289 | -0.15 | -0.16 | 1.145 | 0.710000 |
| docker_006 | NET | 2020-07-19 22:29:00 | 0.93 | 12.0 | 1297 | 0.12 | 0.25 | 0.483 | 1.450000 |
| docker_006 | NET | 2020-07-19 22:30:00 | 10.66 | 135.0 | 1173 | 9.73 | 10.46 | 0.870 | 12.250000 |
| docker_006 | NET | 2020-07-19 22:31:00 | 20.25 | 249.5 | 1232 | 9.59 | 0.80 | 5.795 | 3.490000 |
| docker_006 | NET | 2020-07-19 22:32:00 | 18.65 | 383.0 | 1267 | 8.40 | 0.41 | 35.455 | 1.850000 |
| docker_006 | NET | 2020-07-19 22:33:00 | 36.52 | 455.0 | 1246 | 7.87 | 0.27 | 24.450 | 1.490000 |
| docker_006 | NET | 2020-07-19 22:34:00 | 17.01 | 464.5 | 1260 | 0.33 | 0.01 | 32.183 | 1.240000 |
| docker_006 | NET | 2020-07-19 22:35:00 | 33.50 | 478.0 | 1427 | -3.53 | -0.10 | 36.773 | 0.910000 |

基于基差点 基于移动均值

挑战3：要求故障检测及定位速度快

- 1、故障发生时要求快速发现故障
- 2、本次竞赛规则，对效率要求高

多尺度根因定位

- 1、在10秒内，可以发现大部分故障特征
- 2、如果10秒没发现，放宽至20秒，30秒
- 3、该分钟结束时，再进行一次分钟级计算

优化调用链处理算法

- 1、优化处理调用链
- 2、多进程并行计算
- 3、优化调用链dataframe生成方法

| REASON | TYPE | TIME | PERCENT | NUM | TOTAL | DIFF | DIFF2 | DIFF3 | DIFF4 |
|------------|------|---------------------|---------|-----|-------|-------|-------|-------|-------|
| docker_006 | NET | 2020-07-20 01:41:00 | 8.81 | 8.5 | 118 | 0.00 | 0.00 | 0.00 | 0.00 |
| docker_006 | NET | 2020-07-20 01:42:00 | 7.33 | 1.4 | 85 | -0.14 | -0.14 | -0.14 | -0.14 |
| docker_006 | NET | 2020-07-20 01:43:00 | 1.10 | 0.5 | 105 | 1.40 | 1.40 | 1.40 | 1.40 |

故障发生前

| REASON | TYPE | TIME | PERCENT | NUM | TOTAL | DIFF | DIFF2 | DIFF3 | DIFF4 |
|------------|------|---------------------|---------|-----|-------|-------|-------|-------|-------|
| docker_006 | NET | 2020-07-20 01:41:00 | 8.81 | 8.5 | 118 | 0.00 | 0.00 | 0.00 | 0.00 |
| docker_006 | NET | 2020-07-20 01:42:00 | 7.33 | 1.4 | 85 | -0.14 | -0.14 | -0.14 | -0.14 |
| docker_006 | NET | 2020-07-20 01:43:00 | 1.10 | 0.5 | 105 | 1.40 | 1.40 | 1.40 | 1.40 |

10秒

| REASON | TYPE | TIME | PERCENT | NUM | TOTAL | DIFF | DIFF2 | DIFF3 | DIFF4 |
|------------|------|---------------------|---------|-----|-------|-------|-------|-------|-------|
| docker_006 | NET | 2020-07-20 01:41:00 | 8.81 | 8.5 | 118 | 0.00 | 0.00 | 0.00 | 0.00 |
| docker_006 | NET | 2020-07-20 01:42:00 | 7.33 | 1.4 | 85 | -0.14 | -0.14 | -0.14 | -0.14 |
| docker_006 | NET | 2020-07-20 01:43:00 | 1.10 | 0.5 | 105 | 1.40 | 1.40 | 1.40 | 1.40 |

20秒

| REASON | TYPE | TIME | PERCENT | NUM | TOTAL | DIFF | DIFF2 | DIFF3 | DIFF4 |
|------------|------|---------------------|---------|-----|-------|-------|-------|-------|-------|
| docker_006 | NET | 2020-07-20 01:41:00 | 8.81 | 8.5 | 118 | 0.00 | 0.00 | 0.00 | 0.00 |
| docker_006 | NET | 2020-07-20 01:42:00 | 7.33 | 1.4 | 85 | -0.14 | -0.14 | -0.14 | -0.14 |
| docker_006 | NET | 2020-07-20 01:43:00 | 1.10 | 0.5 | 105 | 1.40 | 1.40 | 1.40 | 1.40 |

1分钟

- 1、实现自适应的阈值调整
- 2、通过DNN、SVM等，提高对已有数据特征内在规律的识别与挖掘率
- 3、面向未来的实际系统，引入大数据流处理等，提升算法的适应性

掘金人

- Real time data is moving
- Anomalies might be shorter than 1 mins
- Data representing the situation might be delayed
- Cut the 1 mins API calling data into 10s interval
 - o Evaluate on success parameter
 - o Elapsed time of a single node
 - o The processing time of each nodes matches a certain trait
- After the 1 mins interval
 - o Will evaluate on this 1 min interval
 - o Using (OSB) data to train a XGBOOST classification model, real time classification
 - o Using the "success" parameter in Service Data to help in analyzing

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99% (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, 调用链数量与异常调用链数量, 异常比例, 异常节点调用次数, 平均时间, 异常率

| startTime | min | max | num | succes_num | elapsed_time | count_per_sec | avg_time | succes_rate |
|-----------------|-----|-----|-----|------------|--------------|---------------|----------|-------------|
| 7/16/2020 22:00 | 0 | 46 | 632 | 632 | 414288 | 13.73913043 | 0.7135 | 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 | 18.625 | 0.6425 | 1 |

```

params = {'learning_rate': 0.2,
          'n_estimators': 75,
          'max_depth': 3,
          '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}
  
```

- First Submit - compare the anomaly to current known anomaly
- Second Submit
 - o Processed call features
 - o Compare processed one with the characteristics of usual features
 - o Use KMEANS to get the node that most probably have a problem
 - o Using RRCF model to do feature matching with the normal platform data
 - o Second Submit
- Most Submit will trigger a checking if the submit meets a threshold
- Use K=1 KNN algo to get the matching characteristic of data then get its type (Root Cause)
- Keep the call trait when running normally, get its different with the data obtained during anomalies occurred, Then use KMEANS Clustering to get the node that has a highest probability of having an anomaly for RRCF input
- ▶ 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_001 | db_007 | db_009 | docker_001 | docker_002 | docker_003 | docker_004 | docker_005 | docker_006 | docker_007 | docker_008 |
|--------------------|-----|--------|--------|--------|--------|--------|------------|------------|------------|------------|------------|------------|------------|------------|
| 7/16/2020 22:36:00 | 0 | 0 | 10 | 128 | 11 | 96 | 45 | 145 | 10 | 0 | 0 | 142 | 0 | 0 |
| 7/16/2020 22:36:10 | 10 | 0 | 0 | 107 | 3 | 58 | 54 | 86 | 9 | 19 | 2 | 115 | 0 | 0 |
| 7/16/2020 22:36:20 | 20 | 0 | 0 | 157 | 22 | 111 | 24 | 212 | 36 | 25 | 0 | 187 | 0 | 0 |
| 7/16/2020 22:36:30 | 30 | 0 | 0 | 118 | 85 | 121 | 29 | 286 | 0 | 0 | 0 | 140 | 0 | 0 |
| 7/16/2020 22:36:40 | 40 | 0 | 5 | 76 | 16 | 112 | 21 | 366 | 16 | 8 | 0 | 18 | 0 | 0 |
| 7/16/2020 22:36:50 | 50 | 0 | 0 | 120 | 44 | 196 | 27 | 330 | 29 | 0 | 2 | 144 | 0 | 0 |

1分钟内, 多次提交, 但时置信水平

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)

pred 0.99071056
judge false traces ['os_018']
rrcf: [['os_018', 'Sent_queue']]
rrcf total time is : 0:00:06.69891

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

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

亚信科技

调用链根因节点定位

算法原理介绍

算法 1 逐节点搜索异常节点
输入: trace调用链, Tree节点树
输出: Tree

```

function RctSearch(trace, parentnode)
if parentnode is None then
parentnode = trace[0]
Tree = Tree.add(parentnode)
end if
chilnode = parentnode.next()
Tree = Tree.add(chilnode)
if condition1 then
Tree = Tree.add(chilnode, 异常)
end if
if condition2 then
Tree = Tree.add(chilnode, 疑似异常)
end if
return Tree
end function
  
```

非疑似异常

```

condition1: elapsedtimeremote > 51% elapsedtimelocal
  
```

疑似异常

```

condition2: elapsedtimeremote - elapsedtimelocal ≥ 51%
  
```

Trace calling chain and tree node tree

Output Tree

1. Anomaly
2. Seeming Anomaly - every node takes time



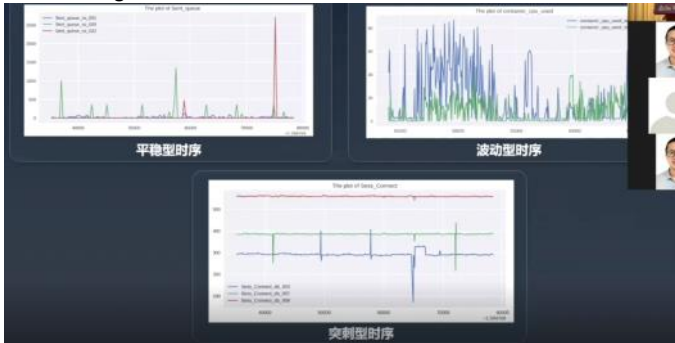
Input: Tree Nodes

Output: node

1. Check every level

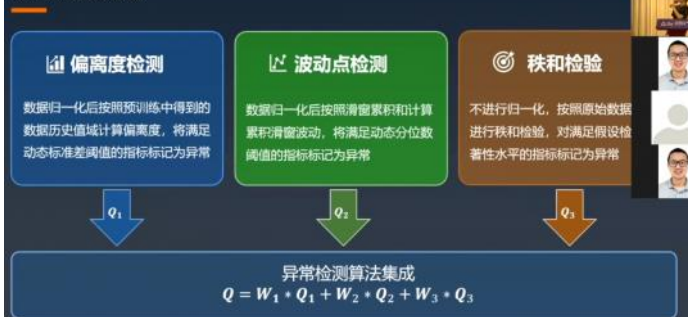
2. $\text{condition: } \frac{\text{Tree}[\text{depth}][\text{status}]}{\sum \text{Tree}[\text{depth}][\text{status}]} \geq 90\%$

Different Algo for different wave

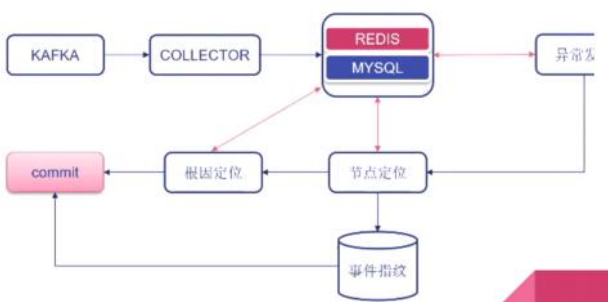


平台指标异常检测

异常检测算法集成



微众银行



1. If there are similar historical event, give the root cause
2. Or else locate the node then lo

事件指纹

Tanimoto系数: $E_j(A, B) = \frac{A \cdot B}{\|A\|^2 + \|B\|^2 - A \cdot B}$ (A 当前异常事件向量 B 对比异常

节点id(onehot)

异常节点上下游

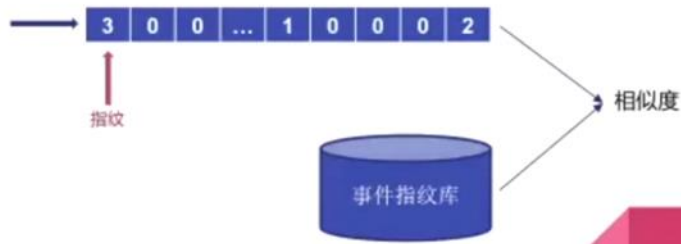
链路存在中断

存在共同上层容器

时延上涨

成功率下降

.....



4. cate the root cause

模型介绍

尝试过的模型

➤ 逻辑回归

· 可解释性强 速度快 实际表现一般

➤ 随机森林

· 实际表现稳定 泛化能力好 速度较快

➤ LSTM

· 实际表现一般 速度较慢

➤ DNN

· 泛化能力强 实际表现最好

