# 数字转型 架构演进

SACC

## 2019 中国系统架构师大会

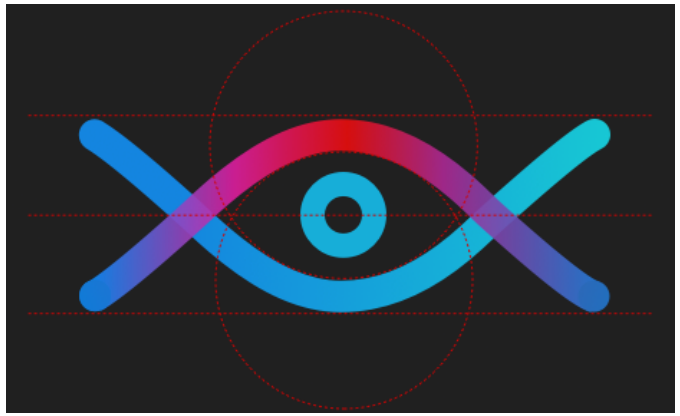
SYSTEM ARCHITECT CONFERENCE CHINA 2019

IT168.com ChinaUnix.net ITPUB

2019年10月31-11月2日 | 北京海淀永泰福朋喜来登酒店

# 大规模时间序列分析与根因定位在苏宁的实践

苏宁科技集团云计算研发中心监控云&AIOps研发中心



出品人：汤泳总监

2019/11/02

# 个人介绍



汤泳，苏宁科技集团云计算研发中心监控云研发中心总监，AIOps研发中心总监，Band级别: 11。

- 出生年月：1978年08月23日
- 工作经验：15年
- 专业：数学与计算机科学系毕业
- 文化程度：硕士
- 领域：海量数据分析、基于深度学习的时间序列分析与预测、自然语言处理、图模型/图神经网络
- 电话：13813896216
- 邮箱：mark.tang19780823@outlook.com

## 业务场景

**数据**

- 基础设施监控
- 实时日志分析
- 端侧监控/调用链

**分析**

- 流式处理
- 多维数据分析
- 时间序列数据分析

**算法**

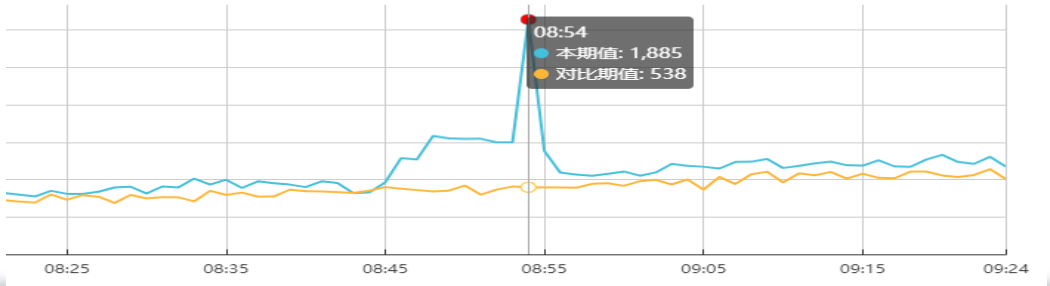
- 时间序列预测
- 异常检测
- 基于图模型的根因分析

# 主要内容

- 背景介绍
- 大规模时间序列分析
- 根因定位
- 异常检测平台深度剖析
- 未来规划

# 背景介绍：Operational 预测




参考文献： Tim Januschowski. (2017). Forecasting at Amazon Problems, Methods and Systems

- Example: Demand forecast for retail products
- millions of time series per scientists (machine learning & software development engineers)
- forecast horizon: days, weeks, at most months
- runs at least daily/on-demand
- hands-off approach
- models can be more black box as long as they are robust
- low counts, bursty, short history and life cycles, intermittent

# 背景介绍：智能异常检测



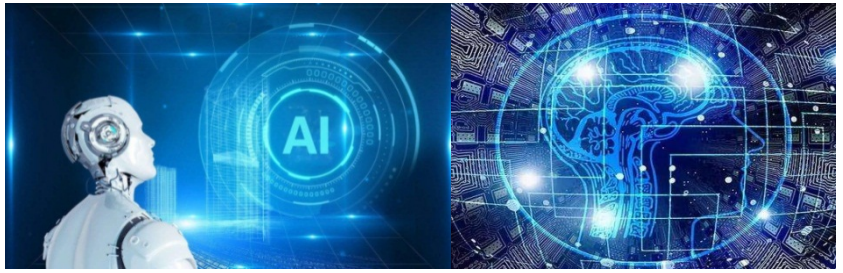
## 传统检测


**挑战**

①面对海量运维监控数据，需要快速止损，人工决策时间往往是小时级但人肉监控(例如ELK)不现实，决策时间往往是小时甚至天级别。

②对于异常点往往需要丰富的经验去识别，但是随着时间的推移，业务数据的特点会发生变化，从而过去的经验也需要与时俱进的更新。

## 智能异常检测




**我们的方法：**

①AI取代缓慢易错的人力决策部分，快速发现问题并且给出决策建 议（分钟级）或提前规避故障。

②使用历史数据结合AI算法自动更新业务经验知识。

# 主要内容



- 背景介绍
- **大规模时间序列分析**
- 根因定位
- 异常检测平台深度剖析
- 未来规划

# 大规模时间序列分析



## 传统时间序列预测方法

**指数平滑（ETS）：**

$$\hat{z}_{T+h} = \alpha z_T + \alpha(1-\alpha)z_{T-1} + \alpha(1-\alpha)^2 z_{T-2} + \cdots + (1-\alpha)^T \hat{z}_1$$

$\alpha$ ：平滑系数

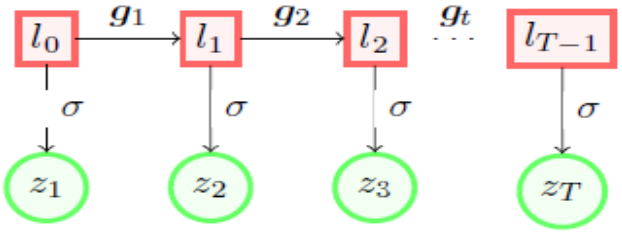
**自回归（AR）：**

$$z_t = \sum_{l=1}^{p} w_l z_{t-l} + b + \epsilon_t$$

$\epsilon_t$ ：高斯噪声 $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$

**状态空间模型（SSM）：**

$$z_t = \boldsymbol{a}_t^T \boldsymbol{l}_{t-1} + \epsilon_t, \quad \epsilon_t \sim N(0, \sigma^2)$$
$$\boldsymbol{l}_t = \boldsymbol{F}_t \boldsymbol{l}_{t-1} + \boldsymbol{g}_t \epsilon_t, \quad \boldsymbol{l}_0 \sim N(\boldsymbol{\mu}_0, \mathrm{diag}(\sigma_0^2)).$$




**传统时间序列预测方法存在的问题：**

- 针对单个时间序列建模，不能充分利用时间序列之间的相关性
- 每个时间序列需要足够的历史数据进行训练
- 本质上只能捕捉线性关系，而不能捕捉非线性关系
- 无法处理时间序列的冷启动问题

# 大规模时间序列分析

## 基于深度学习的大规模时间序列预测方法--DeepAR



### 算法原理及流程：

$$\ell_{\mathrm{NB}}(z|\mu,\alpha)=\frac{\Gamma(z+\frac{1}{\alpha})}{\Gamma(z+1)\Gamma(\frac{1}{\alpha})}\left(\frac{1}{1+\alpha\mu}\right)^{\frac{1}{\alpha}}\left(\frac{\alpha\mu}{1+\alpha\mu}\right)^{z}$$

$$\ell_{\mathrm{G}}(z|\mu,\sigma)=(2\pi\sigma^2)^{-\frac{1}{2}}\exp(-(z-\mu)^2/(2\sigma^2))$$

计算loss

网络

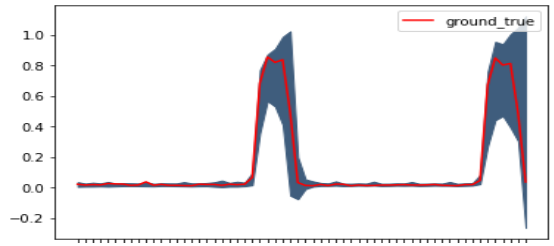
Encoder (LSTM) → Decoder (LSTM)

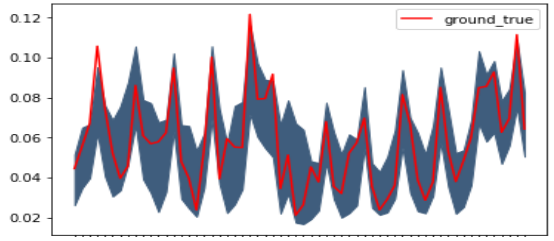
输入

1、上一时刻的真实值
2、当前时刻的特征

1、上一时刻的真实值（训练）/预测值（推理）
2、当前时刻的特征

### 预测效果：




Pros

- 对相关的时间序列建立统一的预测模型，适用于海量数据场景
- 可以同时进行点预测和概率分布预测
- 冷启动预测，实现少量历史数据预测

Cons

- 没有使用attention机制，LSTM对较长的时间序列可能会出现记忆丢失的问题，无法捕获长周期、季节等信息

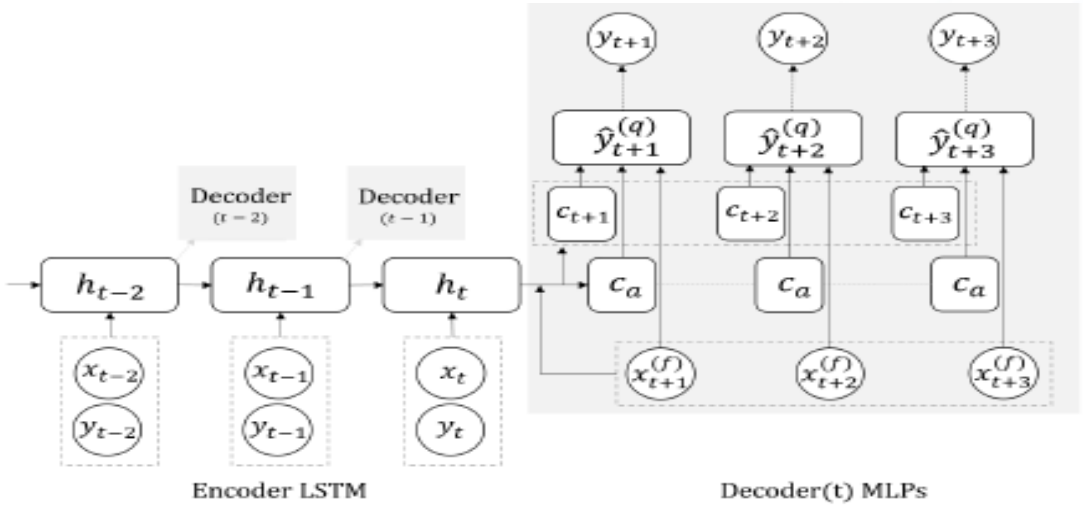
参考文献：Flunkert, V., Salinas, D., Gasthaus, J., and Januschowski, T. (2017). Deepar: Probabilistic forecasting with autoregressive recurrent networks. International Journal of Forecasting, arXiv:1704.04110.

# 大规模时间序列分析

## 基于深度学习的大规模时间序列预测方法--MQRNN



### 算法原理及流程：




Global_MLP

$$(c_{t+1}, \cdots, c_{t+K}, c_a) = m_G(h_t, x_{t:}^{(f)})$$

Local_MLP

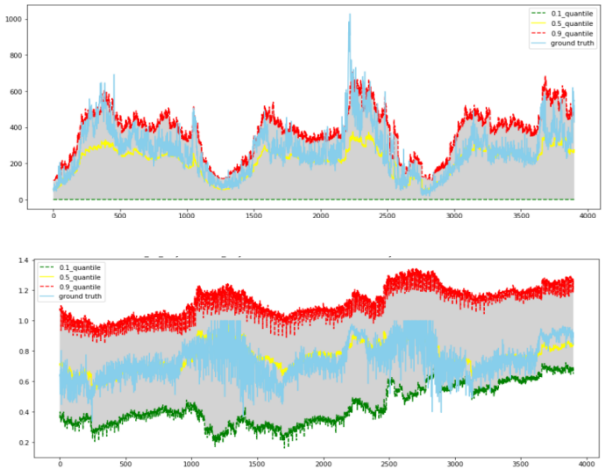
$$(\hat{y}_{t+k}^{(q_1)}, \cdots, \hat{y}_{t+k}^{(q_Q)}) = m_L(c_{t+k}, c_a, x_{t+k}^{(f)})$$

计算loss

$$L_q(y, \hat{y}) = q(y - \hat{y})_+ + (1 - q)(\hat{y} - y)_+$$

$$\sum_t \sum_q \sum_k L_q(y_{t+k}, \hat{y}_{t+k}^{(q)})$$

### 预测效果：



### Fork decoder机制

(1)MQRNN在训练时，Encoder每一个时间点的输出都进行Decoding，loss基于所有Decoder的输出计算；

(2)由于采用了分位数回归机制，Decoder每个时间点的输出与前一个时间点的输出无关，消除了累积误差的影响。

(3)MQRNN输出为分位数矩阵，可以同时得到不同分位数上的预测值。

参考文献： Wen, R., Torkkola, K., and Narayanaswamy, B. (2017). A multi-horizon quantile recurrent forecaster. NIPS Workshop on Time Series, arXiv:1711.11053.

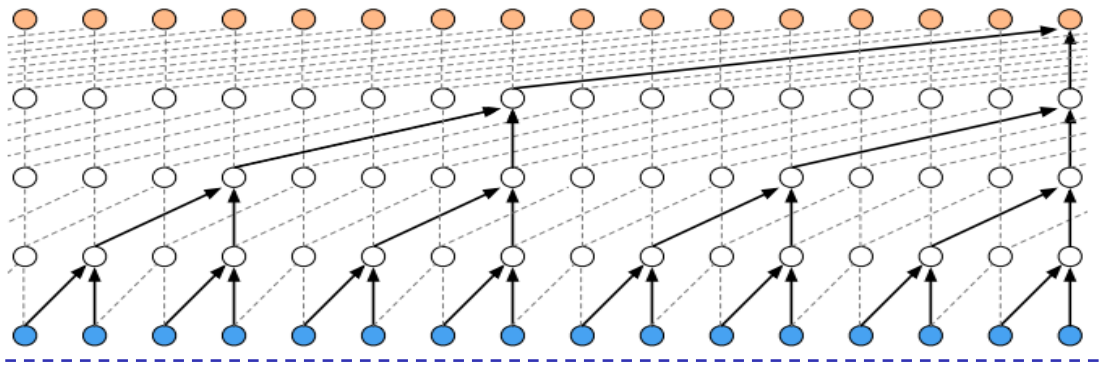
# 大规模时间序列分析

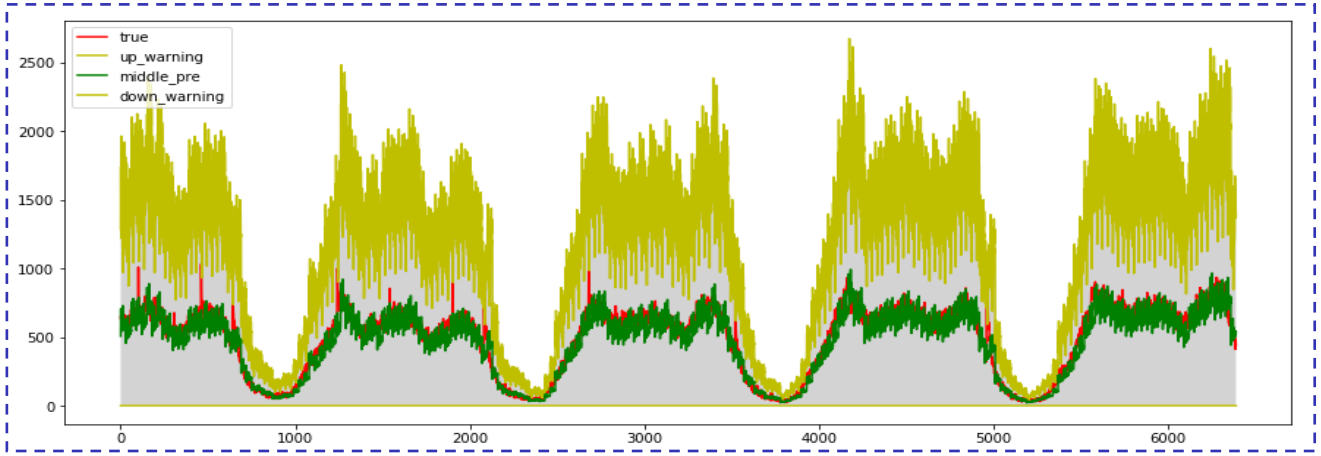
## 基于深度学习的大规模时间序列预测方法--MQCNN



- MQCNN的Encoder采用类似Wavenet的膨胀CNN，decoder与MQRNN一致

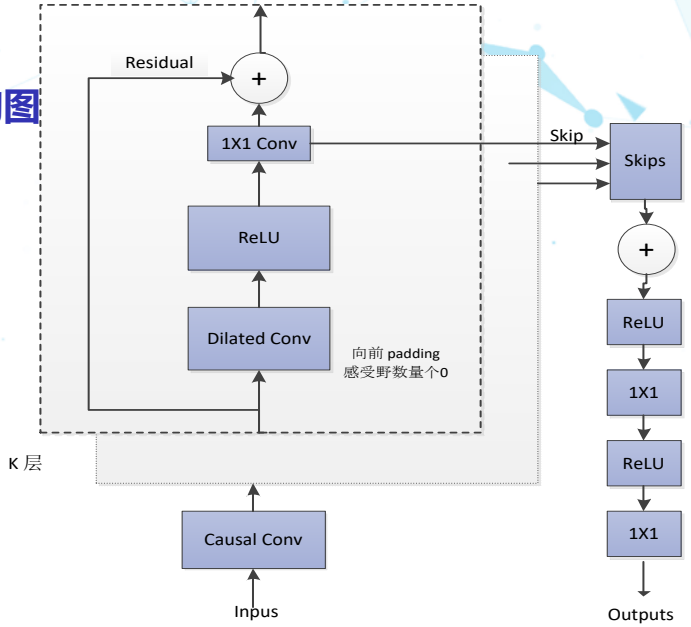
**Encoder部分示意图**




**Wavenet结构图**

使用Dilated Conv可以使CNN的接收域呈指数级增长




**Pros**

- MQC的encoder为CNN，训练速度更快
- 通过Dilated Conv能使模型处理更大长度的输入数据

**Cons**

- 对率型数据预测效果没有计数型好

参考文献： Wen, R., Torkkola, K., and Narayanaswamy, B. (2017). A multi-horizon quantile recurrent forecaster. NIPS Workshop on Time Series, arXiv:1711.11053.

# 大规模时间序列分析

## 基于深度学习的大规模时间序列集成预测方法

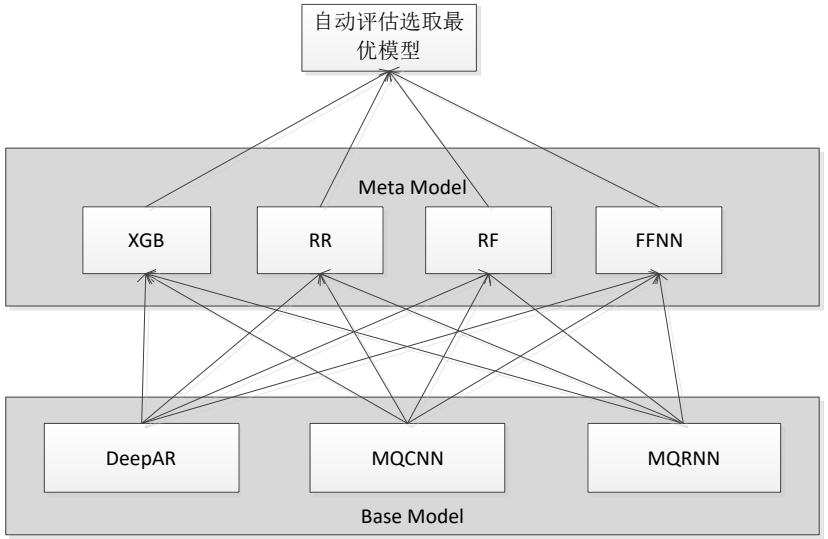


| Count指标 | FFNN_ensemble | XGB_ensemble | RF_ensemble | RR_ensemble | SWA_ensemble | deepar | mqrnn | mqcnn |
|---|---|---|---|---|---|---|---|---|
| average_rmse | 246.5465463 | 215.2504627 | 242.4357493 | 222.3948193 | 222.6 | 449.5706847 | 248.0498881 | 236.7997323 |
| average_smape | 0.23845897 | 0.168967407 | 0.17394335 | 0.183429568 | 0.2336 | 0.226921157 | 0.260262193 | 0.209053173 |

| Rate指标 | FFNN_ensemble | XGB_ensemble | RF_ensemble | RR_ensemble | SWA_ensemble | deepar | mqrnn | mqcnn |
|---|---|---|---|---|---|---|---|---|
| average_rmse | 0.06104917 | 0.061365402 | 0.061787037 | 0.061437497 | 0.0835 | 0.08324926 | 0.207378284 | 0.188028418 |
| average_smape | 0.05566648 | 0.055913178 | 0.057515839 | 0.056988203 | 0.0976 | 0.077240894 | 0.328145736 | 0.307668142 |

### 集成模型：

- 单个预测模型在不同类型的数据上性能差异较大，DeepAR模型对rate指标预测效果较好，但count指标预测效果较差，MQCNN对count指标预测效果较好，但rate指标预测效果较差
- 集成模型的目标是结合各模型的优势，得到比单个模型更优、更鲁棒的结果。使用FFNN（前馈神经网络）/XGB（xgboost）/RF（随机森林）/RR（岭回归）/简单加权平均（SWA）等作为stacking meta learner，最终根据评估指标自动选择最优集成模型。



# 大规模时间序列分析

## 基于深度学习的大规模时间序列集成预测方法



### 集成算法的设计思路及流程：

- **SWA 方法**比较简单，只要计算出相应的权重。**XGB/RR/RF/FFNN等方法**则相对复杂一点，需要进行如下处理：
- **数据预处理：**对deepar,mqrnn,mqcnn的预测结果进行标准化，使用（预测值-均值）/标准差的方法。实验表明，标准化后的集成效果好于非标准化的数据。
- **模型输入：**deepar,mqrnn,mqcnn的预测结果，时间序列的特征（分钟，小时，星期几等），时间序列的编号等。
- **损失函数：**均方差，并增加L2正则化，防止过拟合。

### 集成模型自动评估和选择：

- 使用测试数据真实值与集成模型预测值的SMAPE作为评估指标。自动选取测试集上SMAPE值最小的模型作为最终的集成模型。

### 预测上下边界生成：

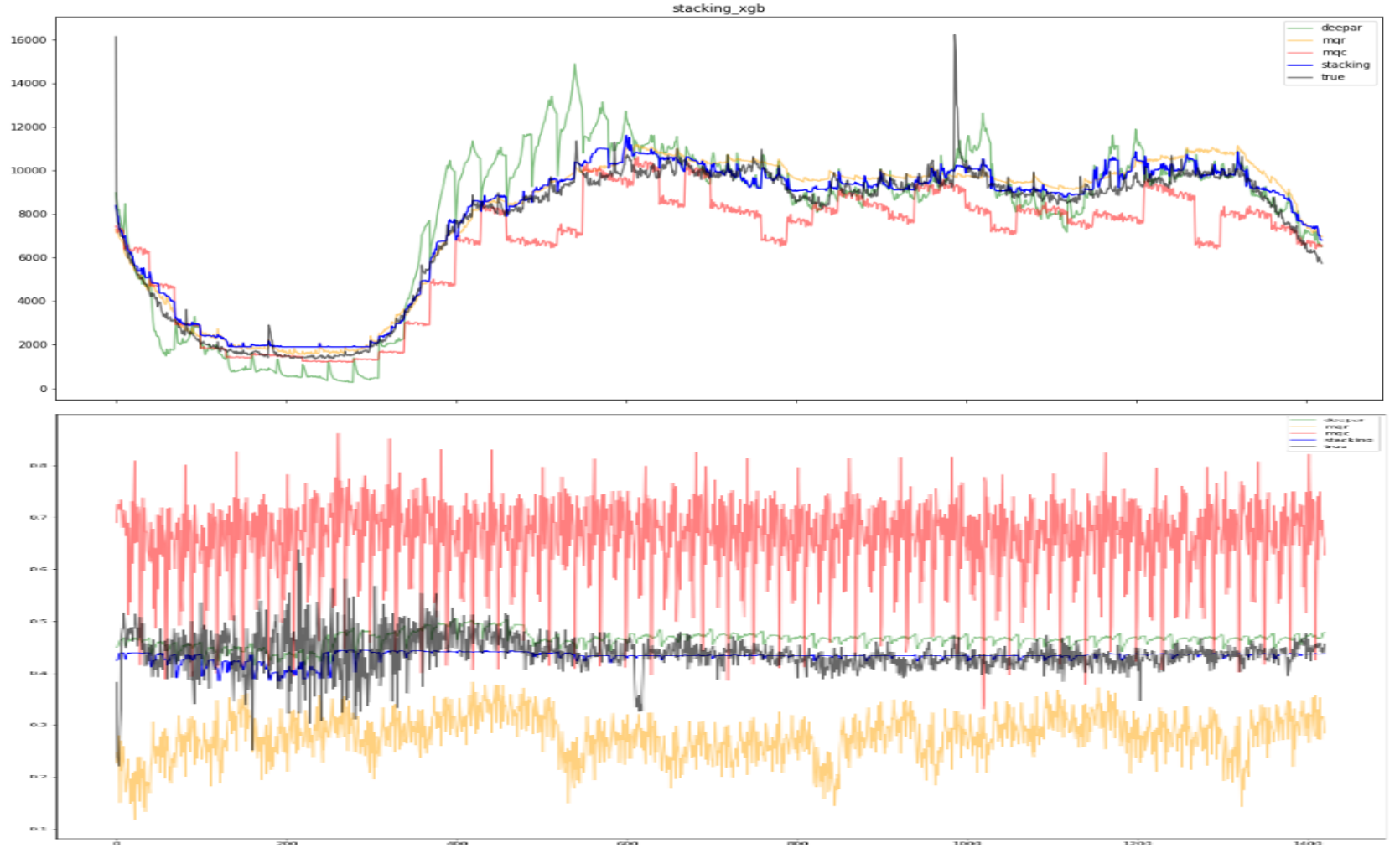
- 使用集成模型的预测值及各基模型的标准差作参数，采样选取合适的分位数生成上下边界。

# 大规模时间序列分析

## 基于深度学习的大规模时间序列集成预测方法



- 上图为count型数据使用xgboost集成的效果。下图为rate型数据使用FFNN集成的效果。
- 黑色为真实值；绿色，黄色和红色分别为deepar,mqrnn和mqcnn的预测值。蓝色为集成后的预测值。
- 集成后的预测值比单个模型的预测值更稳定，更接近真实值。

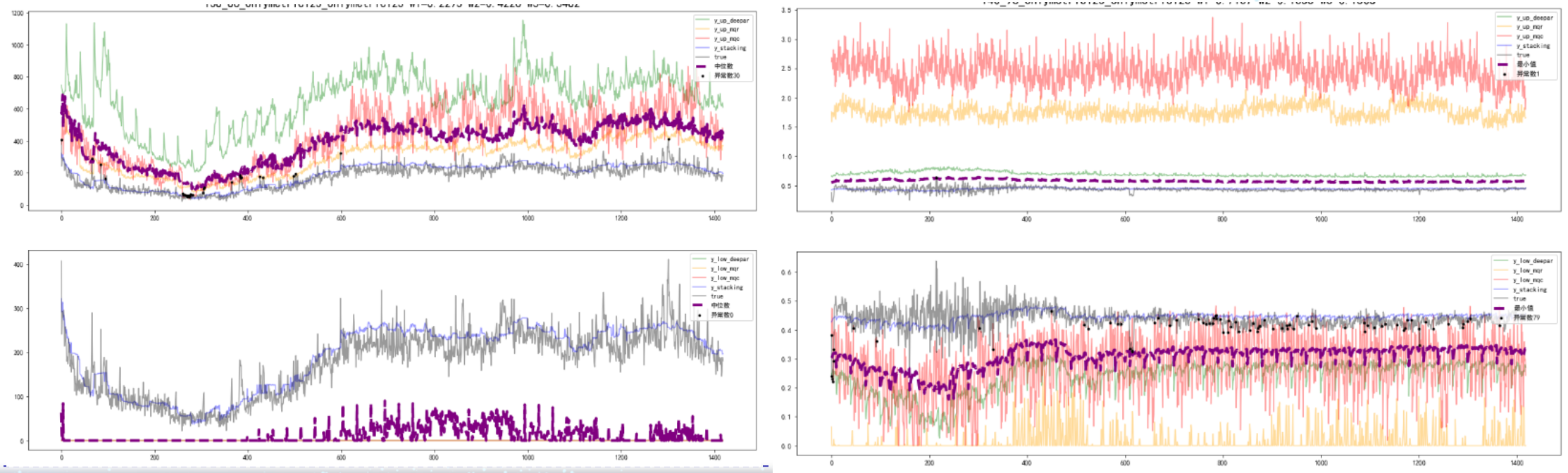


# 大规模时间序列分析

## 基于深度学习的大规模时间序列集成预测方法



- 下图显示的为count类型数据和rate类型数据，根据集成模型预测值，生成上下边界的效果图。
- 其中：黑色为真实值；绿色，黄色和红色分别为deepar,mqrnn和mqcnn的预测上下边界值；蓝色为集成后的预测值；紫色为集成后的上下边界值。
- 由图可知，集成后的预测边界比单个模型的预测边界更稳定，更合理。



# 主要内容



- 背景介绍
- 大规模时间序列分析
- **根因定位**
- 异常检测平台深度剖析
- 未来规划

# 根因定位



## 背景

**可加性KPI（如登录量、支付成功数等）是业务中的一类重要的指标。当可加性KPI总指标发生异常时，我们希望使用根因定位算法快速定位导致总指标出现异常的根源（例如：南京地区的iOS端的家电品类的支付成功数异常导致了总的支付成功数发生异常）。**

一个KPI指标根据多种属性分别监控。比如Page Views（PV），分别统计来自不同ISP和不同省份的PV。如Fig.1 所示：

| $f(p,i) \rightarrow v(p,i)$ | | Province($p$) | | | |
|---|---|---|---|---|---|
| | | Beijing | Shanghai | Guangdong | * |
| ISP ($i$) | Mobile | 20→14 | 15→9 | 10→10 | 45→33 |
| | Unicom | 10→7 | 25→15 | 20→20 | 55→42 |
| | * | 30→21 | 40→24 | 30→30 | 100→75 |

Fig.1

表中的f和v分别表示PV的期望的正常值和观测值，我们看到红色的部分观测值和正常值不符，这些PV就是发生了异常。

根因定位问题希望的是定位发生异常时那些属性取值导致了异常，比如Fig.1中的根因就应该是(Province=Beijing&Shanghai, ISP=*)

主要存在以下挑战：

- 不同维度值组合之间不是相互独立的，根因的异常通常会传播到其他维度值组合，导致不同维度值组合异常的纠缠，真正的根因难于甄别
- 要对异常根因进行定位，须对所有维度值组合构成的巨大空间进行搜索。以登录线为例，数据包含4个维度，总的最细粒度维度值组合数量在百万量级，而搜索空间更是百万量级的指数级别(>2^1000000)。

配图来自： https://www.zhihu.com/question/331456480/answer/806871325

# 根因定位

## HotSpot算法核心思想



针对上述难点，我们探索并利用Hotspot算法作为解决方案。该算法的核心思想包括以下三方面：

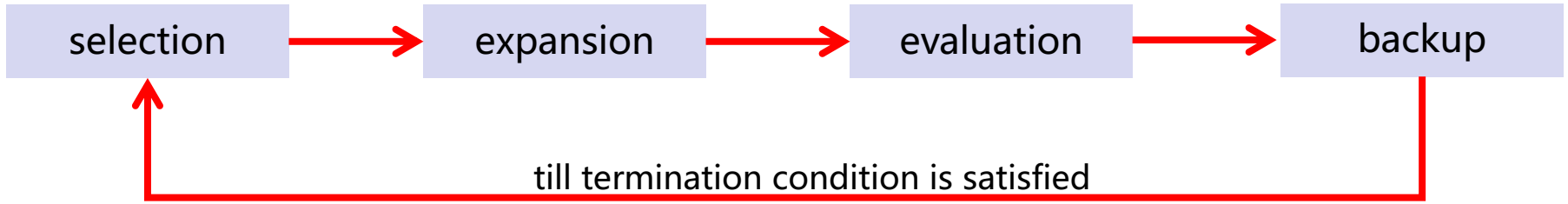
① ripple effect：异常传播模型，用于解决维度值组合之间异常的互相纠缠问题。

$$\frac{\mathrm{v(child)} - \mathrm{f(child)}}{\mathrm{f(child)}} = \frac{\mathrm{v(parent)} - \mathrm{f(parent)}}{\mathrm{f(parent)}}$$

② potential score：用于评估某维度值组合作为根因的置信程度。

$$ps = max(1 - \frac{d(real, deduced)}{d(real, forecast)}, 0)$$

③ 蒙特卡洛树搜索（MCTS）：用于解决在巨大空间中的搜索问题。




参考文献：[1] Y. Sun, Y. Zhao, Y. Su, D. Liu, X. Nie, Y. Meng, S. Cheng, D. Pei, S. Zhang, X. Qu et al., "Hotspot: Anomaly localization for additive kpis with multi-dimensional attributes," IEEE Access, vol. 6, pp. 10 909–10 923, 2018.

# 根因定位

## 根因定位整体流程

■ **数据准备**

1、最细粒度指标预测。

2、依据最细粒度指标预测结果进行预剪枝，剔除大量对总指标异常几乎不构成贡献的维度值，形成预剪枝后的维度值集合。

3、构建全维度值（预剪枝后的）组合的最细粒度指标预测值和真实值。

■ **层次化蒙特卡洛树搜索**

1、对最细粒度指标的预测值和真实值构建所有视角下的数据立方体（cuboid），例如cuboid_city、cuboid_loginType是单维度cuboid、cuboid_city_loginType是一个2维度cuboid，以此类推。

2、依据数据立方体的维度数量对cuboid进行分层，单维度cuboid为第一层，全维度cuboid在最底层。

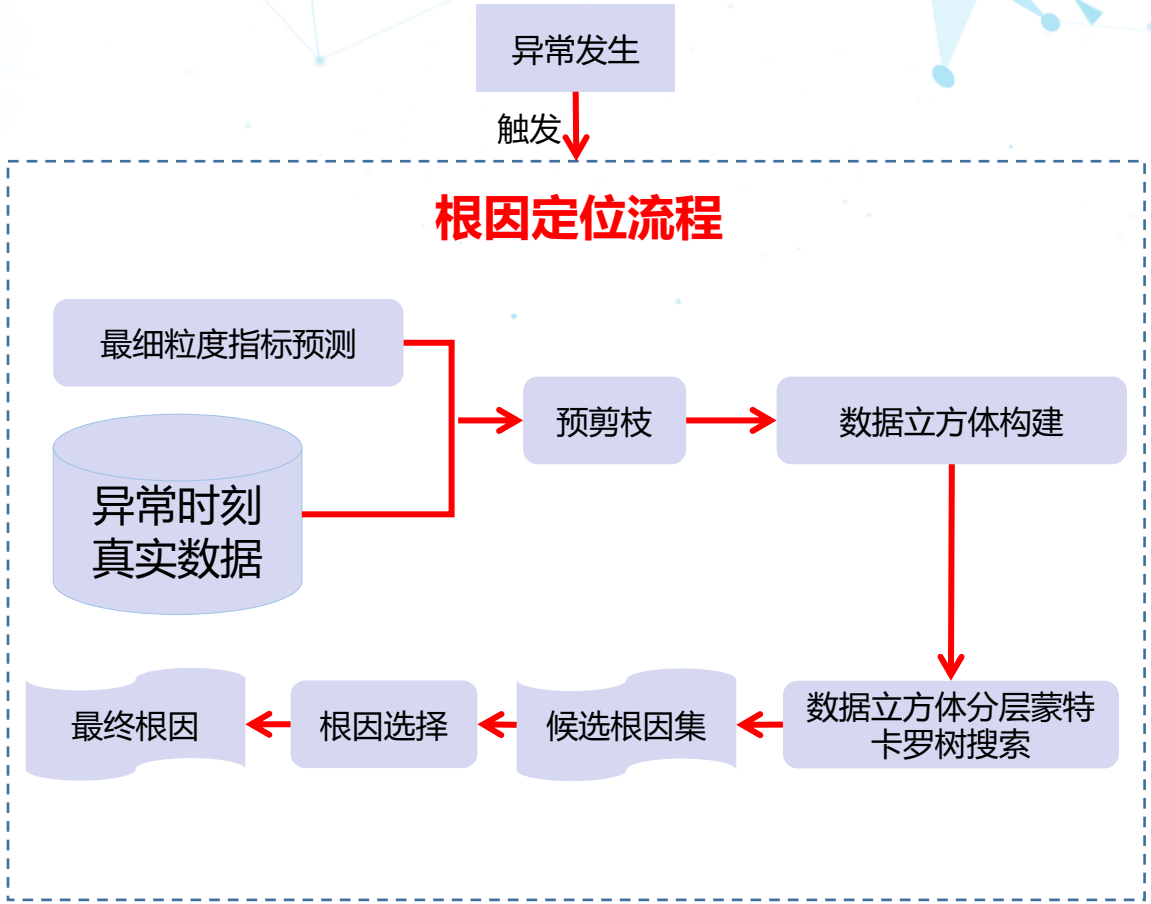
3、自上而下分别对每一层的每一个cuboid进行搜索。利用上层搜索结果对下层进行剪枝后，再进行下层的搜索。搜索过程层间串行，层内并行。

4、每一个cuboid的搜索结果是一个根因元素集，作为最终根因集的候选集。

■ **根因确认**

1、从所有cuboid的搜索结果中选择potential score最大的候选根因集作为最终根因集。

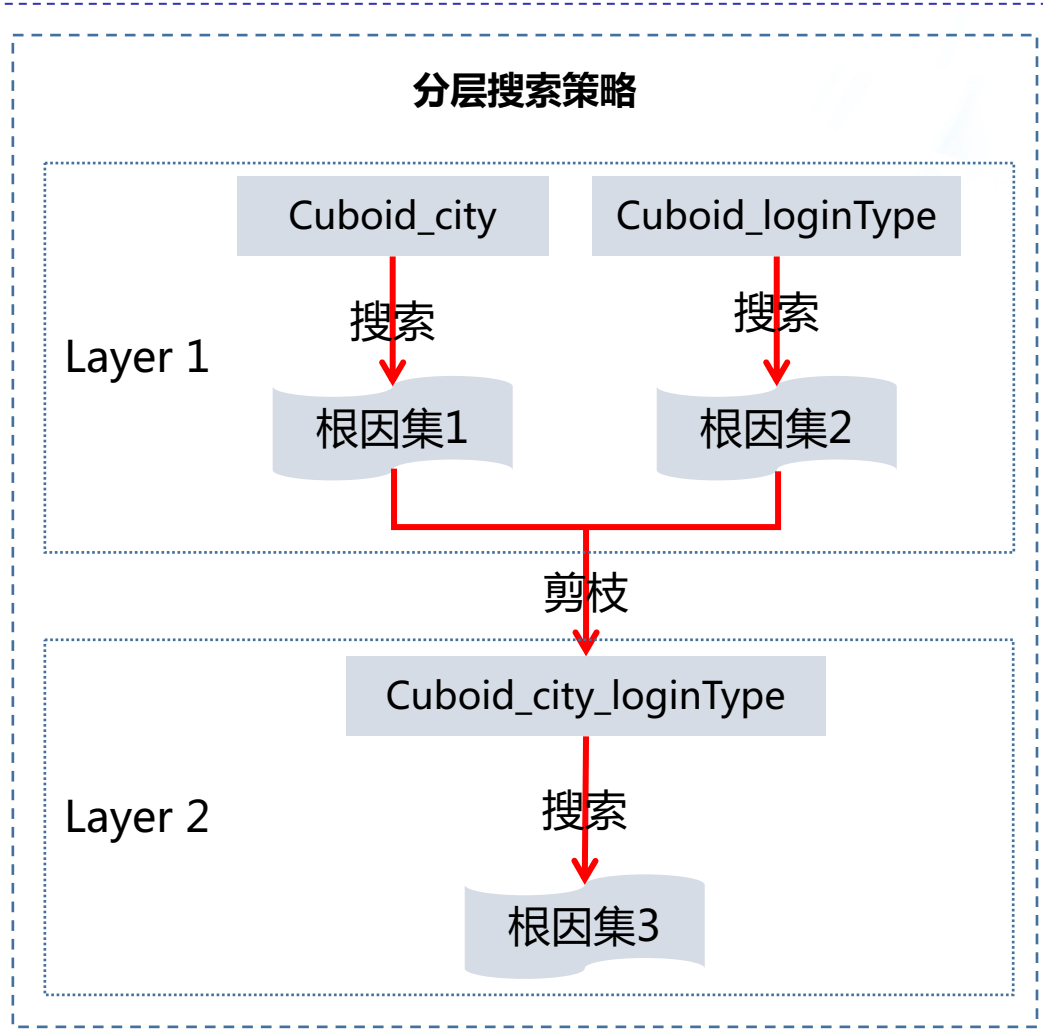
2、如遇两个cuboid的候选根因集的potential score接近，根据奥卡姆剃刀原理选择最简者。

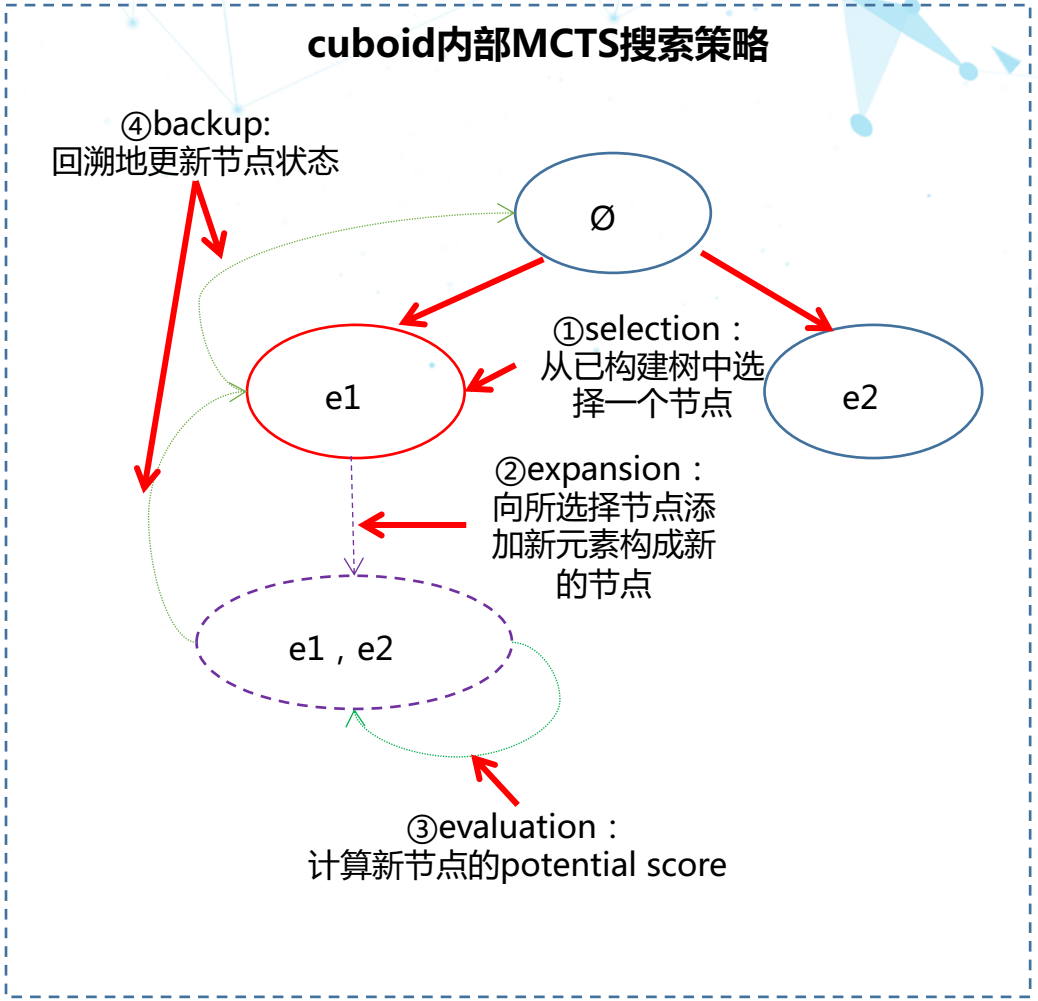


# 根因定位

## 蒙特卡洛树搜索（MCTS）





# 根因定位

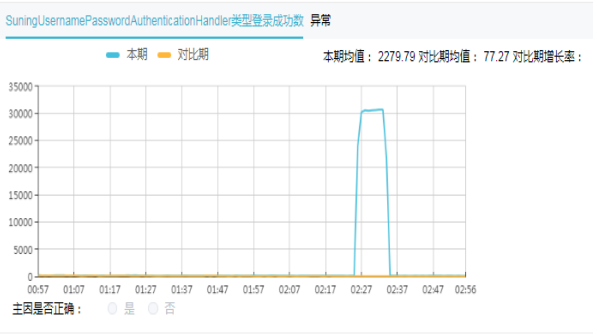
## Hotspot应用案例

某日凌晨02:26，登录成功量发生异常陡增。

Hotspot将根因定位到：{地区: 内网IP, 会员角色: 142000000154，类型: SuningUsernamePasswordAuthenticationHandler}。

如下图，后期分析结果验证了根因定位结果的准确性。左图为发生异常的总指标，右图为定位结果分解后的3个单维度指标。






# 根因定位



## Hotspot算法的演化

**HotSpot存在的问题：**

①只能对基础可加性指标进行根因定位，不能直接处理复合指标，如成功率等。

②容易忽略变化幅度较小的异常。

③定位深层cuboid根因和由多个元素构成的根因时，准确率下降。

④计算量较大，算法运行时间较长。

**考虑到HotSpot存在的问题，Squeeze[1]算法进行改进，Squeeze的主要优势是：**

①提出泛化的ripple effect（GRE），可以直接处理由可加性指标复合得到的指标，如成功率等。

②改进的potential score（GPS）对变化幅度较小的异常也较为敏感，不易忽略此类异常。

③当根因所在层次较深或者根因集包含多个元素时，准确率不发生明显下降。

④不需要构建全量数据，计算量较小，算法运行时间相对稳定。

参考文献：[1] Zeyan Li, Chengyang Luo, Yiwei Zhao, Yongqian Sun et al. “Generic and Robust Localizationof Multi-Dimensional Root Causes”，ISSRE 2019， Berlin, Germany, Oct 28-31, 2019

# 根因定位

## Squeeze算法基本原理

Squeeze算法分为两个主要的环节：通过Bottom-Up Searching缩小搜索空间、通过Top-Down Localization进行根因定位。

■ Bottom-Up Searching

①deviation based filtering：通过寻找绝对偏差的累积概率分布的膝点对叶子元素进行过滤，过滤掉大部分正常叶子元素。

②deviation score based clustering：基于相对偏差，对经过filtering得到的异常叶子元素进行分组，每组代表一个异常簇。

■ Top-Down Localization

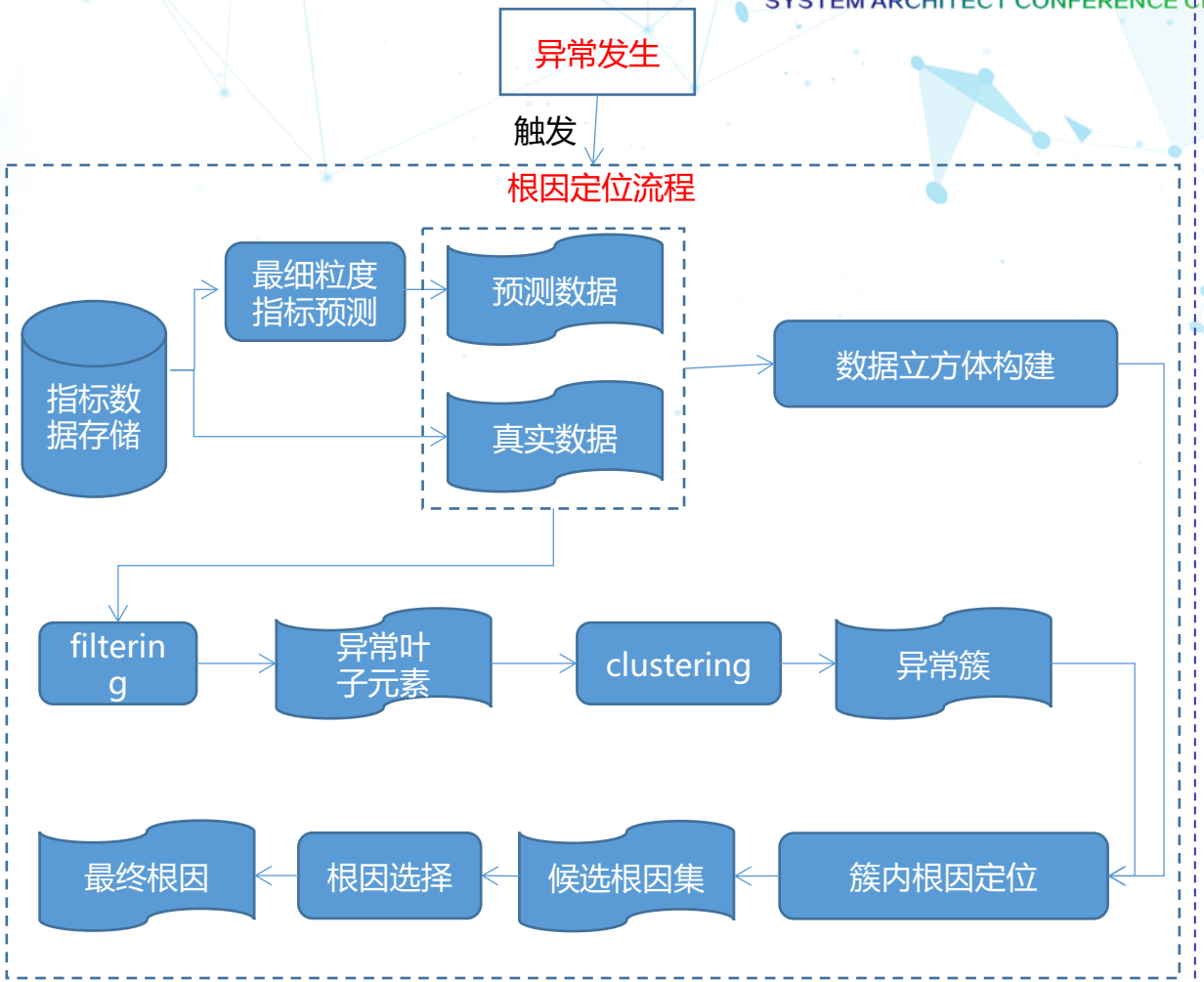
对每一个异常簇，进行簇内的根因定位，输出导致该异常簇的根因集。

簇内根因定位的基本思想是：

①层次化搜索策略：对预先构建的数据立方体（cuboid）进行层次化的搜索，如果上层cuboid搜索结果满足终止条件，即终止搜索。

②cuboid内搜索策略：以descent score作为优先搜索策略，以泛化的potential score（GPS）作为评价指标，定位最大GPS的元素集作为该cuboid的候选根因集。

③根因确认：针对所有cuboid的候选根因集，依据奥卡姆剃刀原理，对候选根因集的GPS和简洁性进行平衡，选择最终根因。




参考文献：[1] Zeyan Li, Chengyang Luo, Yiwei Zhao, Yongqian Sun et al. “Generic and Robust Localizationof Multi-Dimensional Root Causes”，ISSRE 2019， Berlin, Germany, Oct 28-31, 2019

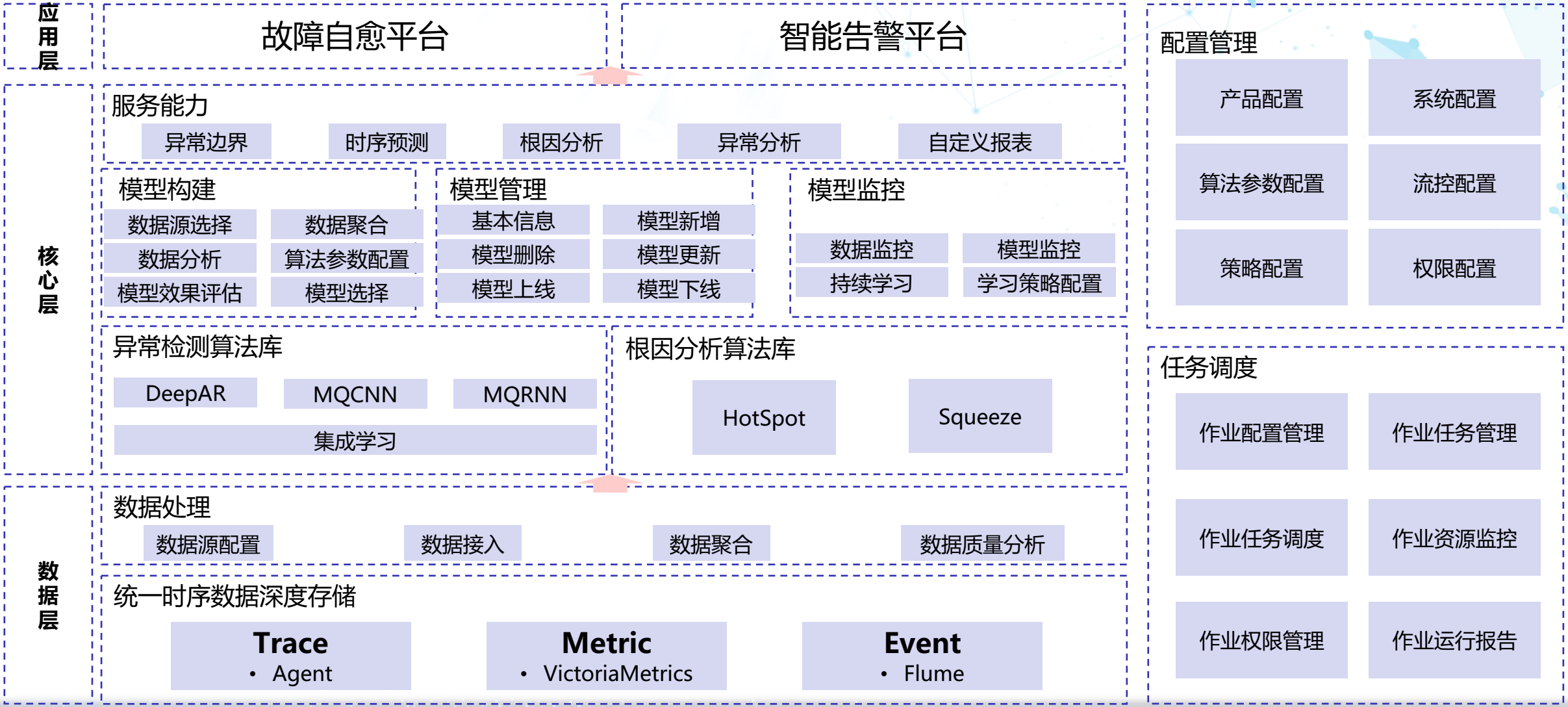
# 主要内容



- 背景介绍
- 大规模时间序列分析
- 根因定位
- **异常检测平台深度剖析**
- 未来规划

# 异常检测平台深度剖析

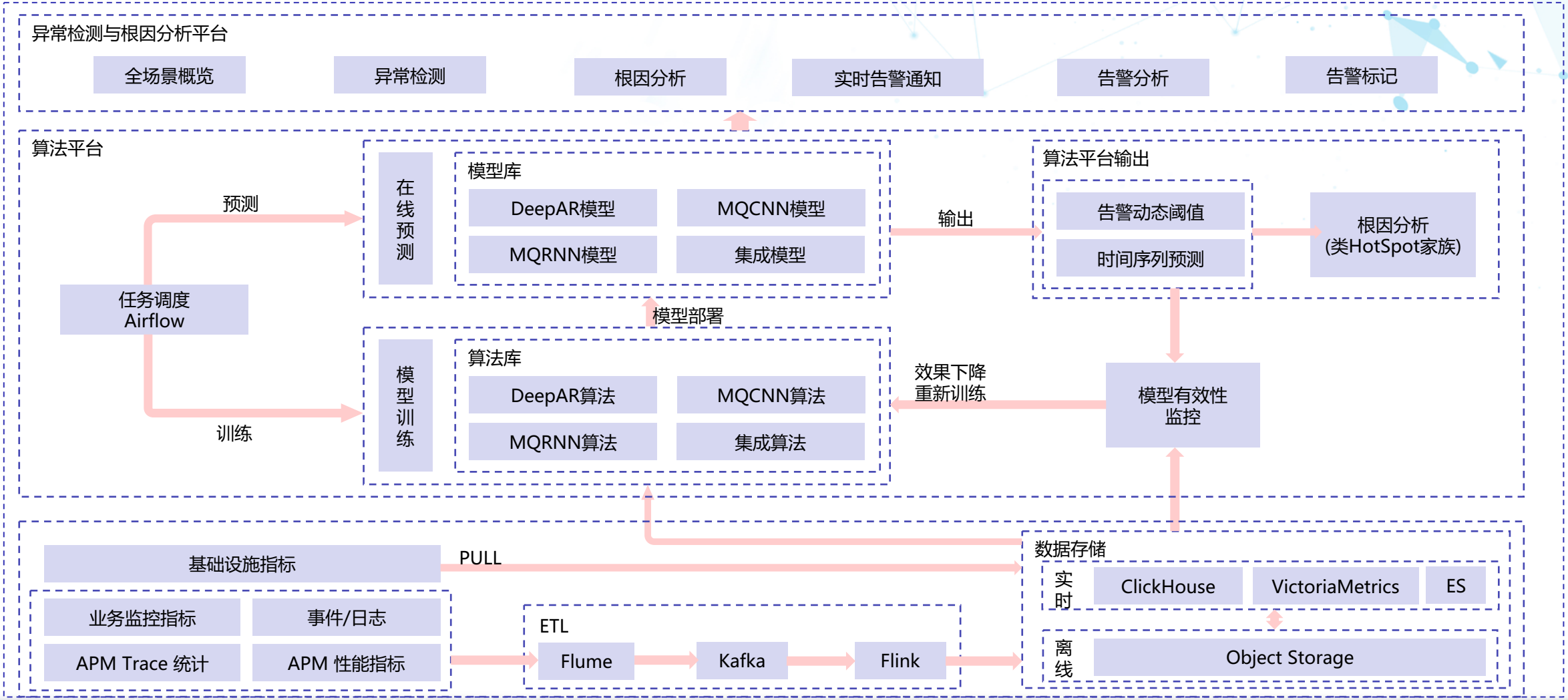
## 平台能力



# 异常检测平台深度剖析



## 平台技术架构



# 异常检测平台深度剖析



## 数据源接入

### 数据接入

实时消息接入 | 实时文件接入 | 数据实时聚合

### 数据存储

实时批量写 | 实时聚合 | 数据离线备份

分布式存储 | 数据统计备份 | 系统信息存储

### 数据分析报告

数据量趋势分析 | 数据延时分析 | 描述性统计分析

- 数据源多样化：支持kafka、clickhouse、VictoriaMetrics(Prometheus家族)等多种数据源的接入
- 数据聚合：可以根据指定的维度和指标对数据进行聚合
- 数据衍生：支持根据不同的指标进行运算，从而得到用户想要的复合数据
- 关键性能指标：目前上限tps为5.8w/s，最高可支持50w/s
- 分布式存储处理：支持数据异步写入处理。目前使用clickhouse集群作为存储介质，使用10台物理服务器进行分布式处理。
- 数据分析：从接入的数据量趋势、接入时间来分析数据的质量，做到从源头对数据进行监控

# 异常检测平台深度剖析

## 模型管道的构建

# 异常检测平台深度剖析



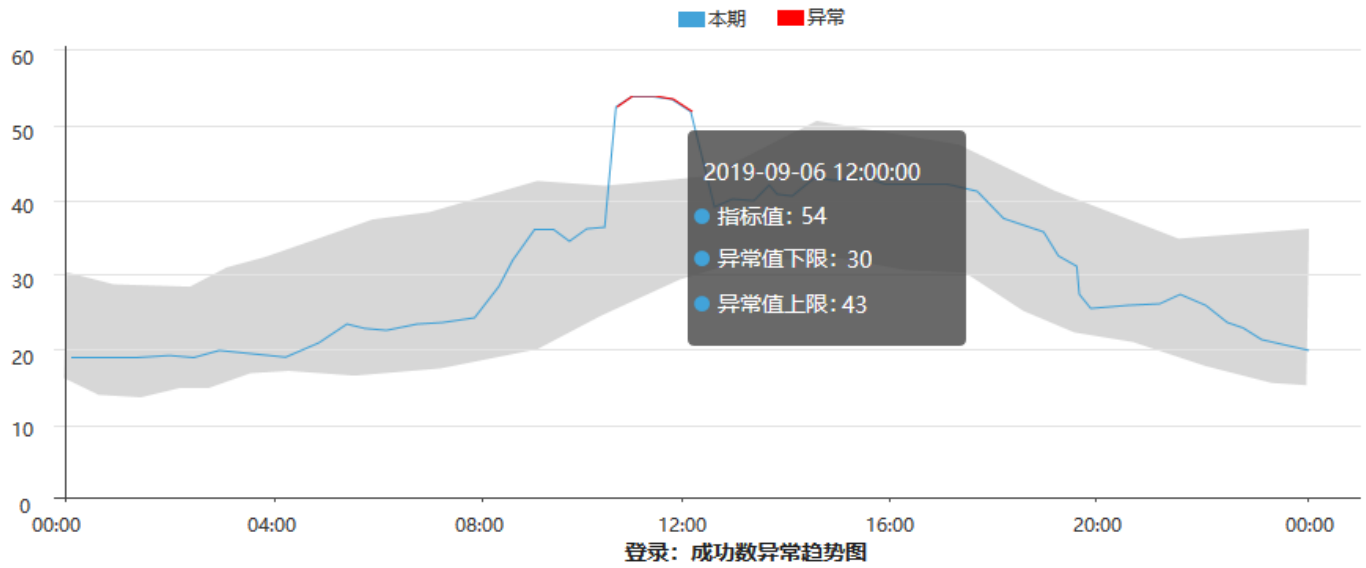
## 异常点检测

### 异常检测流程

1. 使用近400分钟的指标信息，通过构建的模型，预测未来30分钟每1分钟的预测值和概率分布函数。
2. 设置采样率，根据概率分布函数进行随机采样。
3. 定义异常百分率，使用采样后的数据计算自动获取异常上下边界。
4. 指标值与异常点上下边界进行匹配，判断该点是否为异常点。

### 异常检测结果

- 蓝色曲线为指标正常值，红色曲线为检测出来的异常值。
- 灰色阴影部分为异常上下边界阈值。

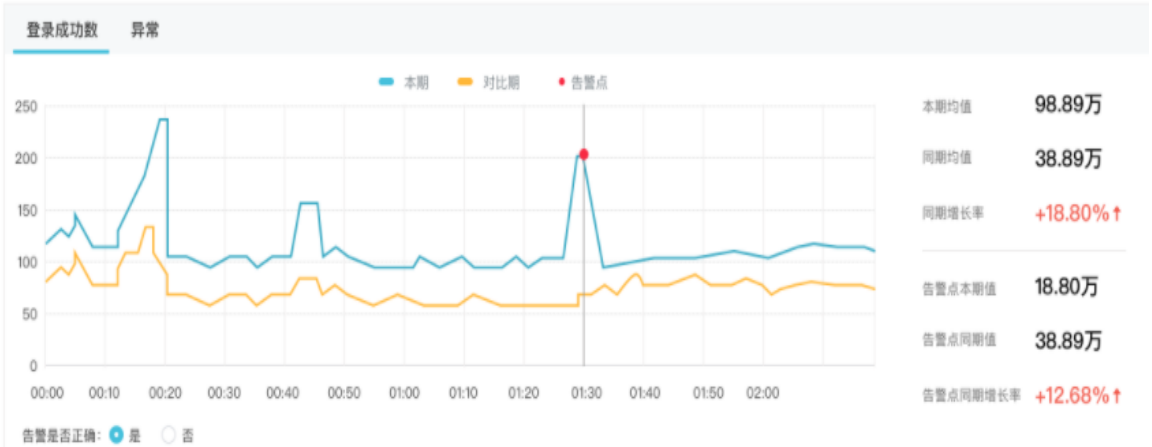


登录：成功数异常趋势图

# 异常检测平台深度剖析



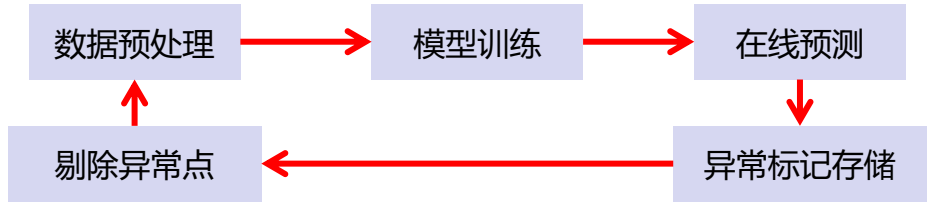
## 告警分析





### ■ 告警趋势分析:

1. 根据告警发生时间进行趋势分析
2. 统计期内指标对比分析，结合业务知识判断告警的准确性
3. 对发生的告警进行标记，为后续模型优化提供数据基础。



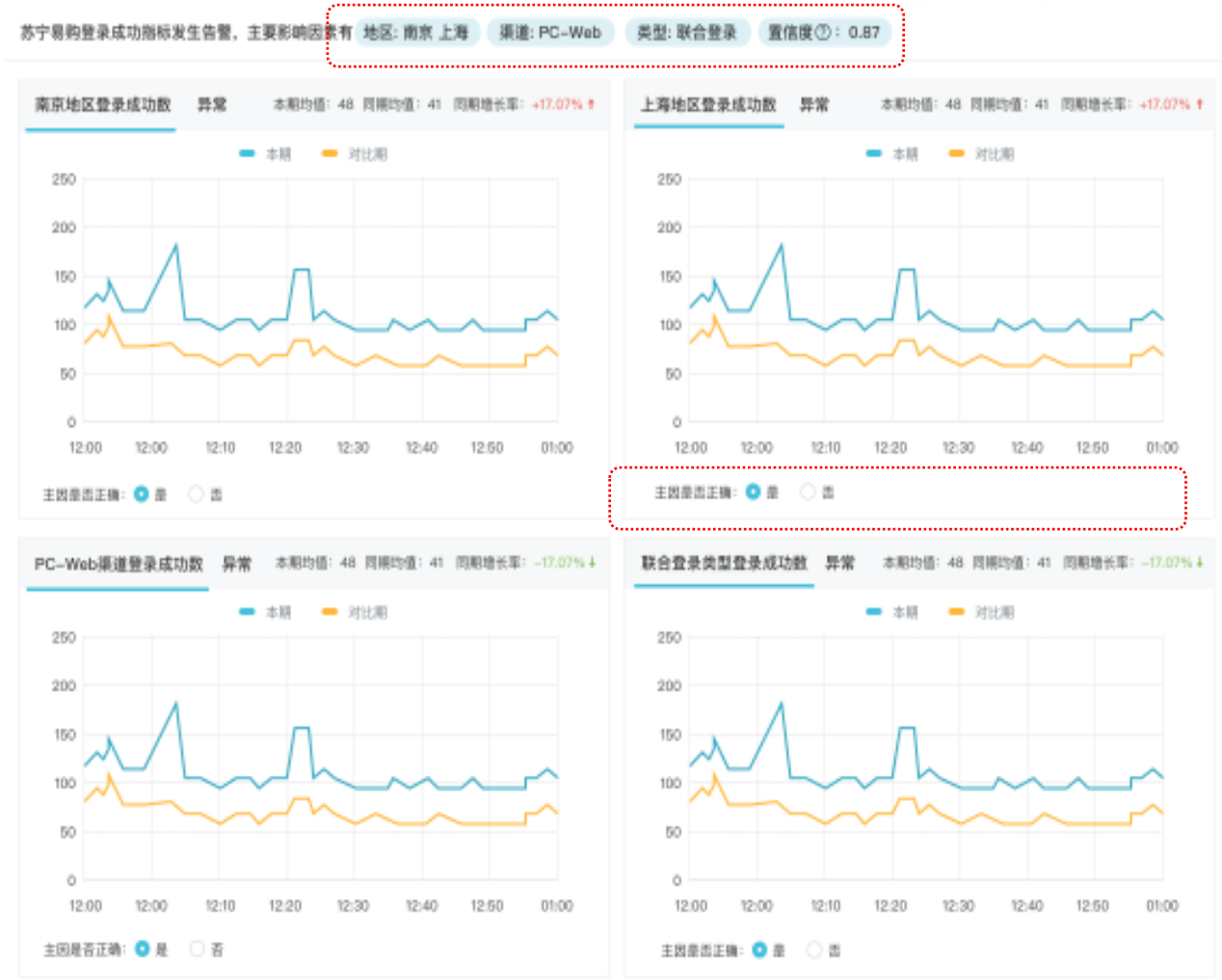

### ■ 告警异常码分析：

1. 为用户提供异常码分析，帮助用户可以快速判断告警的准确性。

2. 用户根据异常码，可以定位异常明细，全方位显示异常信息。。

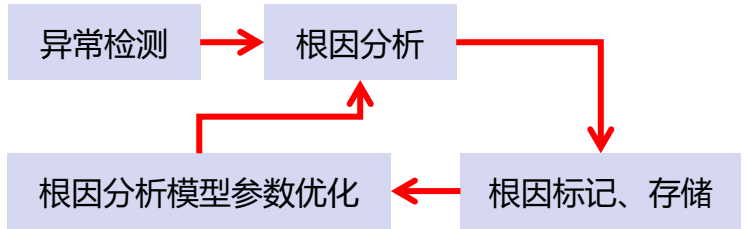
# 异常检测平台深度剖析

## 根因分析




- 影响因素：以单维度指标为基础，使用HotSpot家族的算法对异常数据进行分析，定位出主要的影响因素，帮助用户快速定位异常数据维度，大大提高工作效率；

- 告警标记：用户在告警分析和处理中，根据最终的影响因素来对算法分析出的主因进行判断标记，后续算法根据标记信息进行算法调优，形成良性循环。



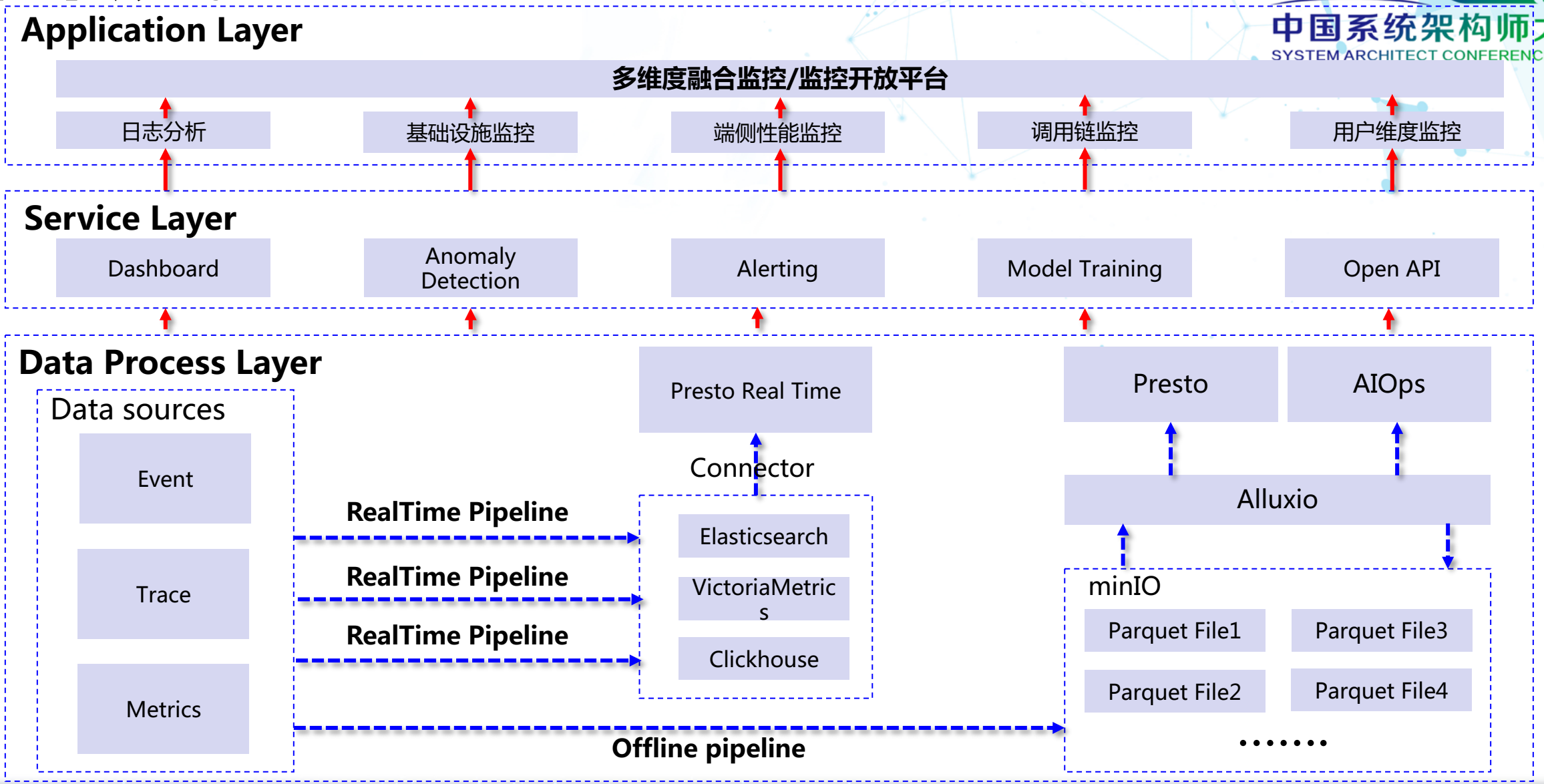
# 主要内容



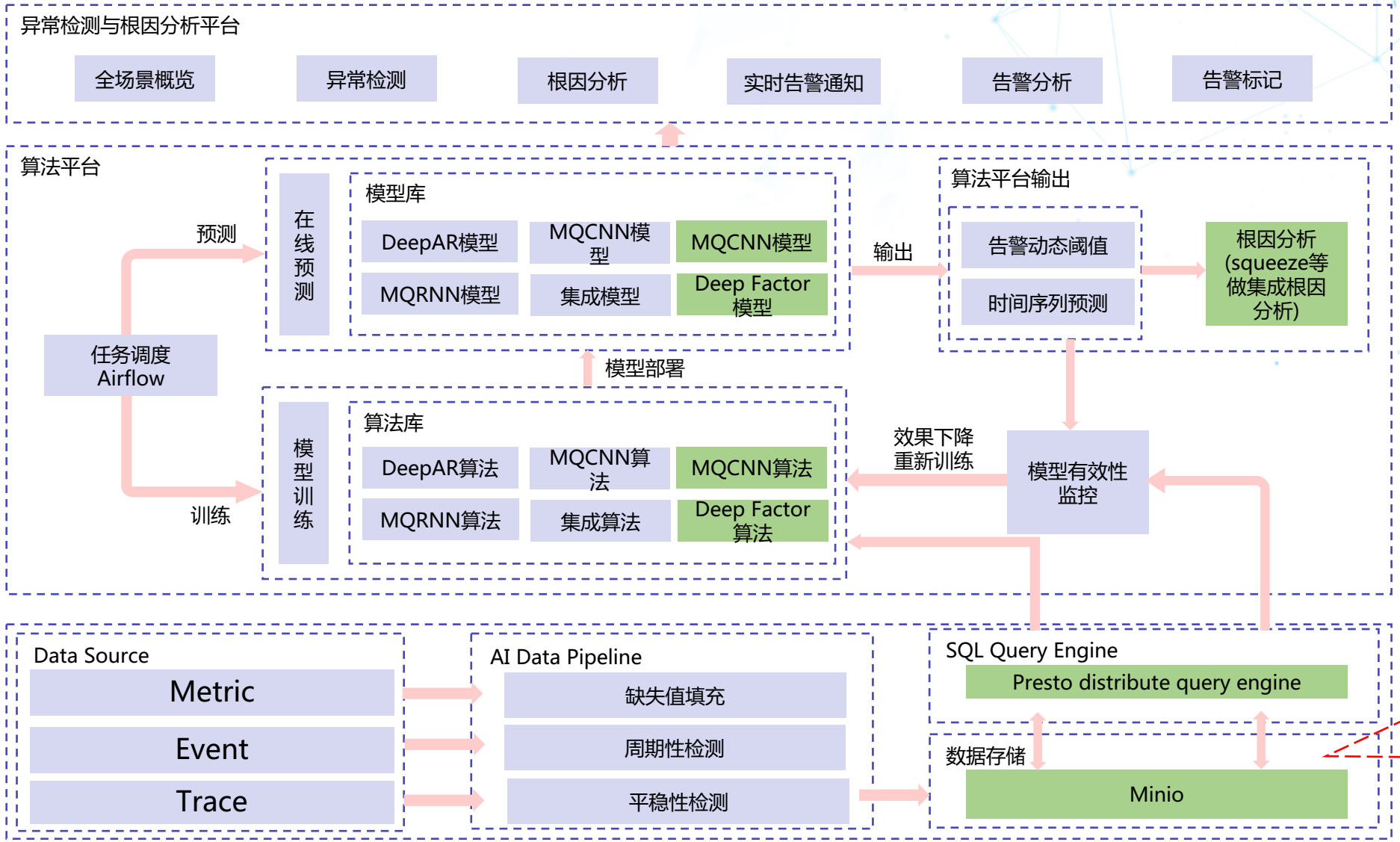
- 背景介绍
- 大规模时间序列分析
- 根因定位
- 异常检测平台深度剖析
- **未来规划**

# 未来规划





# 未来规划




- 自定义仪表盘异常点分析
- TF框架升级到2.0
- 研发类似AWS GluonTS的统一异常检测/根因定位的Library
- 异常检测算法继续拓展（深度状态空间，深度高斯过程）
- 根因定位算法拓展到图神经网络以及图顶点熵领域
- 使用AutoML技术，实现自动调参，模型架构自动搜索功能
- 预测结果以API/SDK方式提供给下游
- 使用MinIO实现统一存储，使用Presto统一分析引擎。

# Thanks