



**QCon** 全球软件开发大会  
INTERNATIONAL SOFTWARE  
DEVELOPMENT CONFERENCE

BEIJING 2017

# 智能运维里的时间序列

异常检测·根源分析·预测

赵宇辰 / 销售易技术VP

# 智能运维里的时间序列

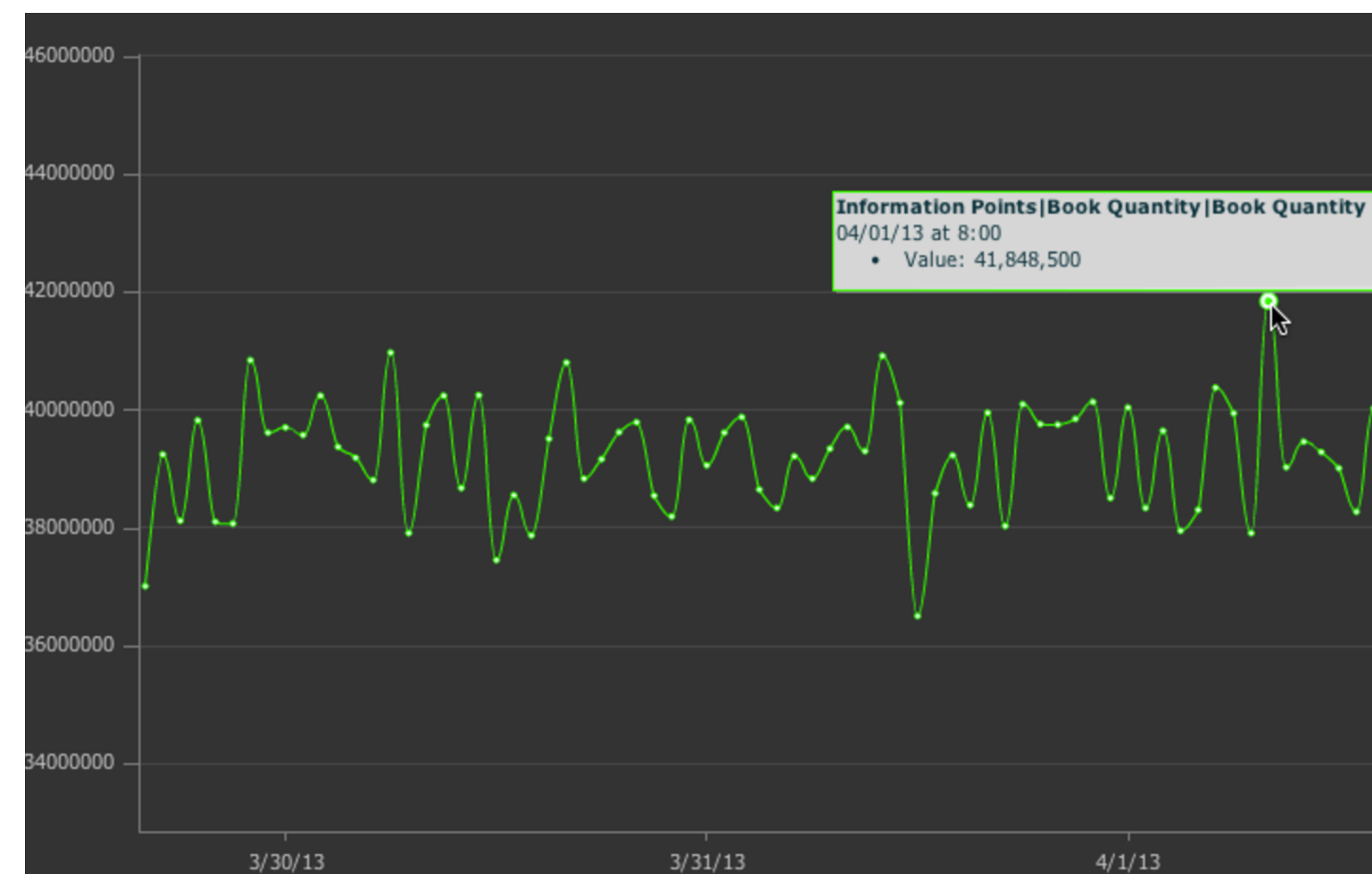
- 性能监控里的时间序列
- 传统方法
- AI + 时间序列
  - 异常检测
  - 根源分析
- 总结



# 性能监控里的时间序列

常见的时序序列 / metrics :

- Block Time (ms), Average Block Time (ms)
- Calls, Number of Calls
- Calls/min, Calls per Minute
- CPU Used (ms), JVM CPU Burnt (ms/min)
- Errors/min, Errors per Minute
- Response Time (ms), Average Response Time (ms), Avg. Time per Call
- Slow Transactions, Number of Slow Calls
- Stalled Transactions, Stall Count
- Wait Time (ms), Average Wait time (ms)



# 性能监控里的时间序列

常见的时间序列类型：

- Backends / 后端
- End User Monitoring / 用户UE
- Mobile / 手机
- Service End Points / 服务端
- Overall Application Performance / 全局
- Business Transaction Performance / 业务相关
- Application Infrastructure Performance / 基础设施
- Errors / 错误





# 性能监控里的时间序列

常见的时序计算：

- 原始数值 ( observation )
- 最小值 / 最大值 ( min / max )
- 总和 ( sum )
- 平均值 ( avg )
- 数量 ( count )
- 百分比 ( % )
- 百分位数 ( percentile )



# 智能运维里的时间序列

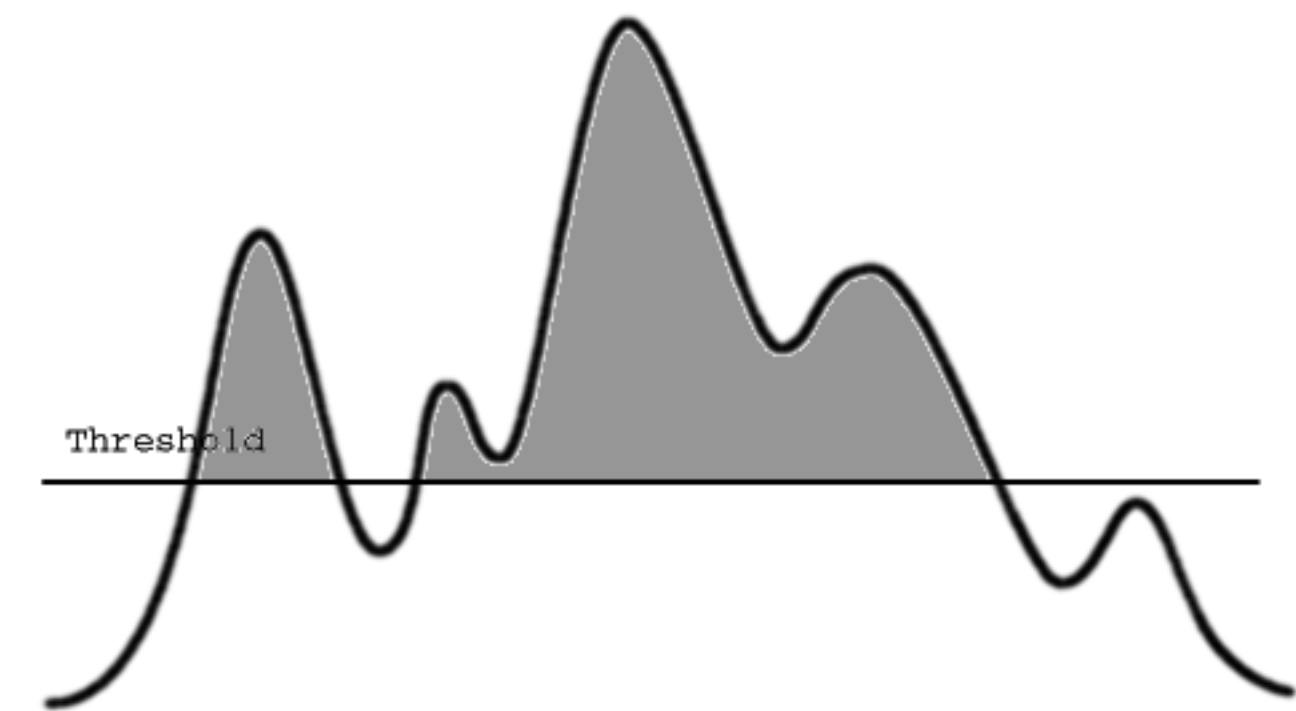
- 性能监控里的时间序列
- **传统方法**
- AI + 时间序列
  - 异常检测
  - 根源分析
- 总结



# 传统方法

## 方法一：固定阈值

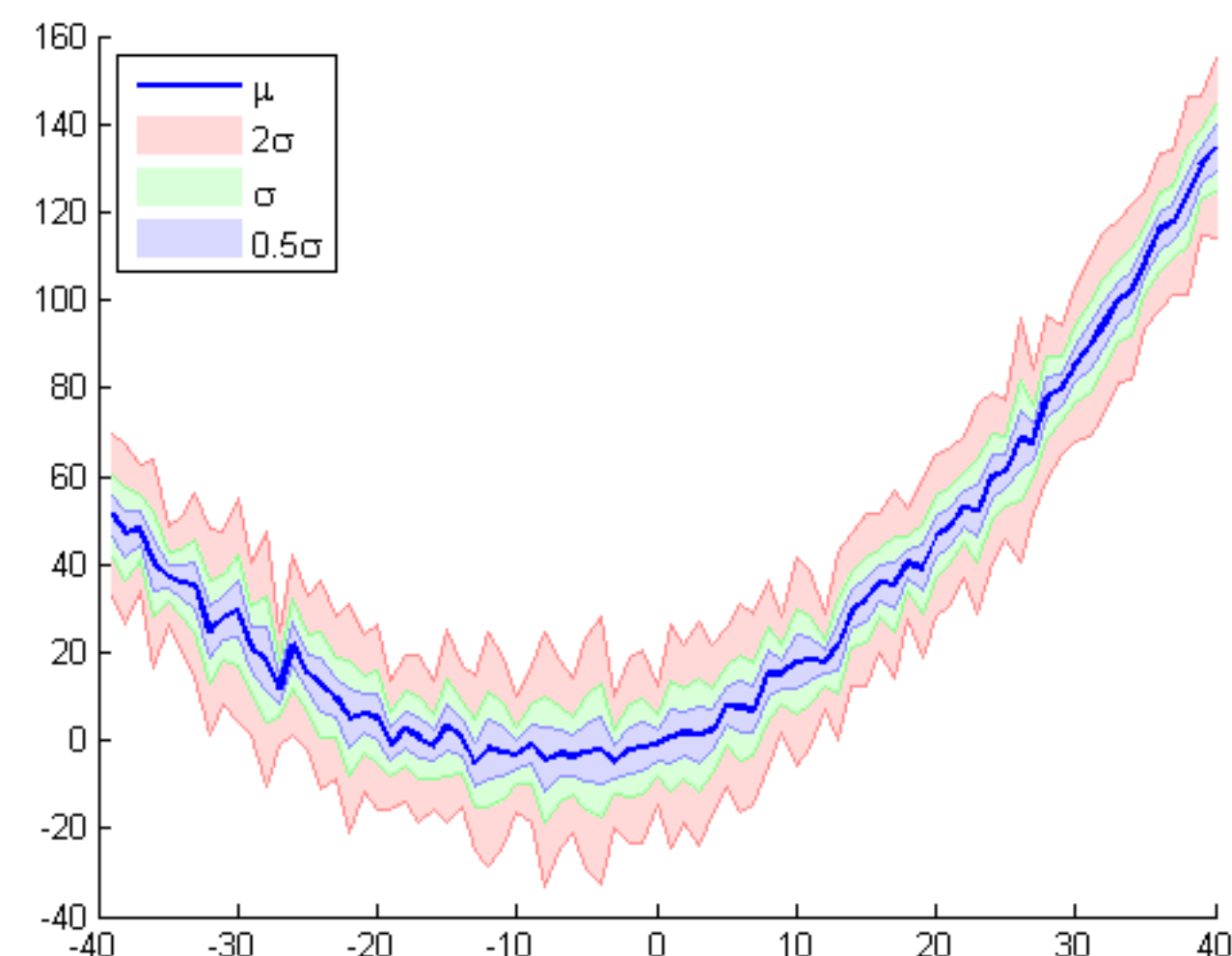
- 如果 $\text{value} > \text{阈值}X$ ，发出警报
- 如果 $\text{value} < \text{阈值}Y$ ，发出警报



# 传统方法

## 方法二：动态阈值

- 计算
  - (1) 平均值  $\mu$
  - (2) 方差  $\sigma$
- 正常取值区间： $\mu \pm 2\sigma$
- 如果value在区间外，发出警报





# 传统方法往往效果不好



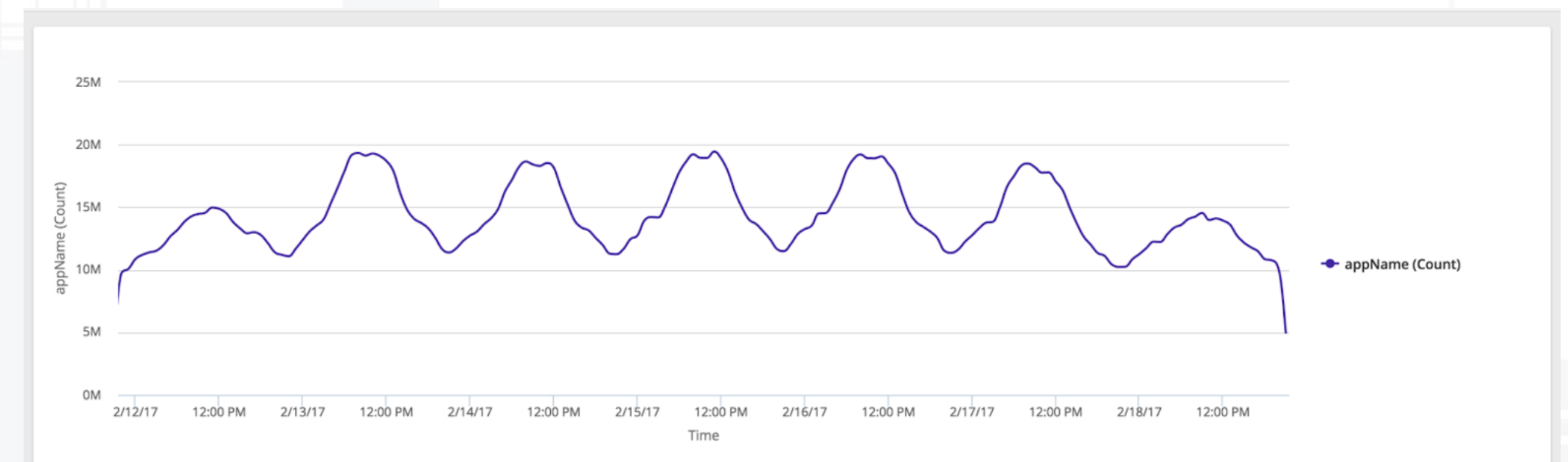
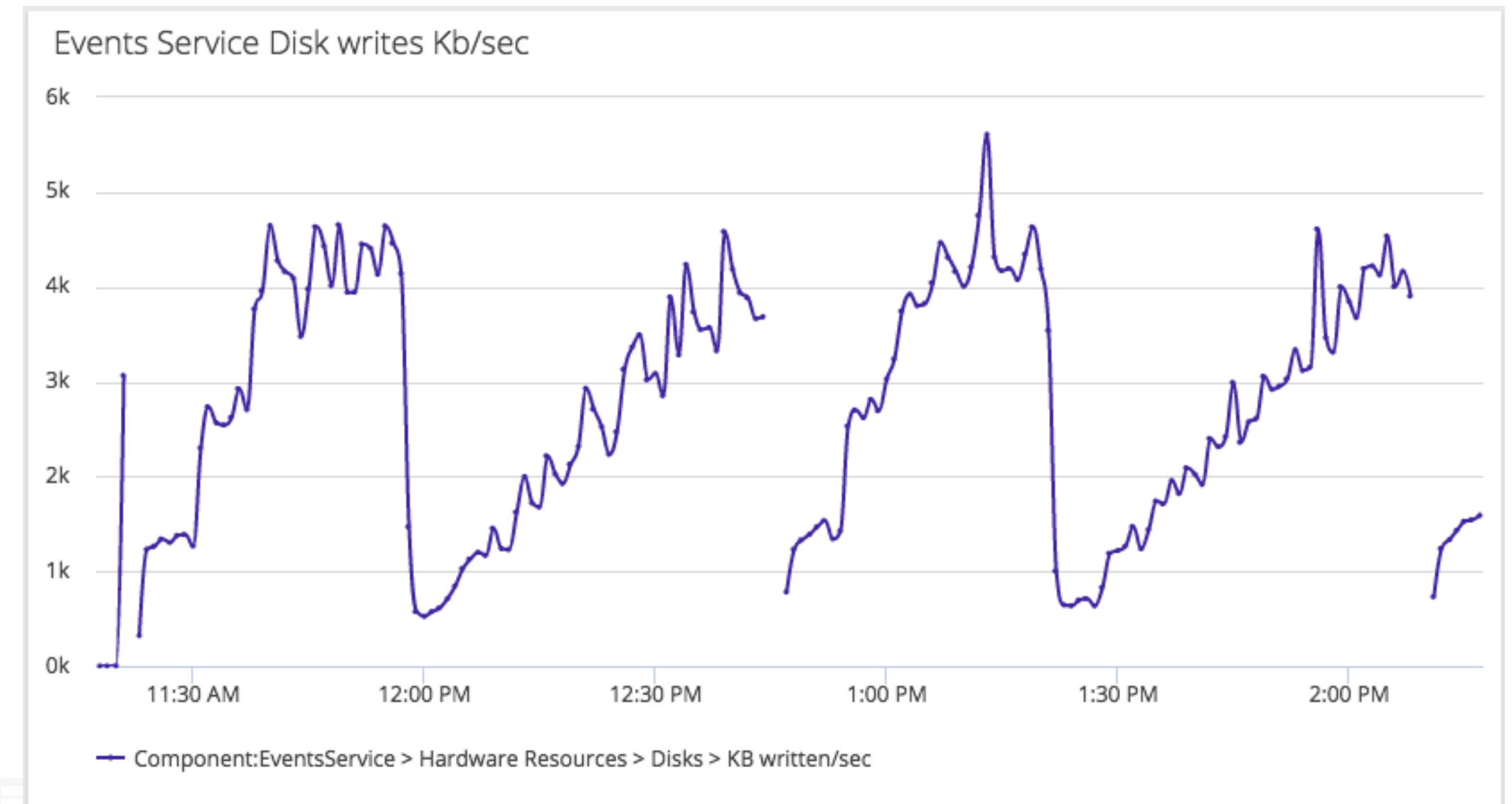
# 原因1: 忽视周期性 ( seasonality )



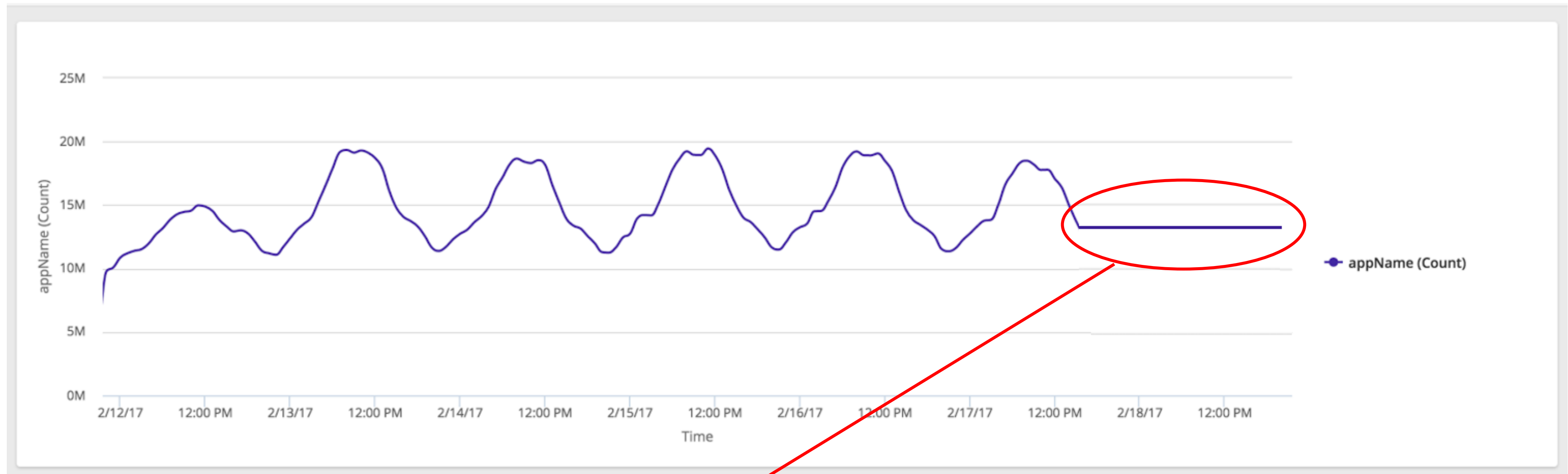
# 周期性的时间序列

例子：

- 每两小时的定时任务（cron job）
- 每两周升级计划



# 周期性的时间序列



在平均值附近，传统方法不会发出警报，  
但是否真的没有异常？

# 原因2: 忽视趋势 ( trend )





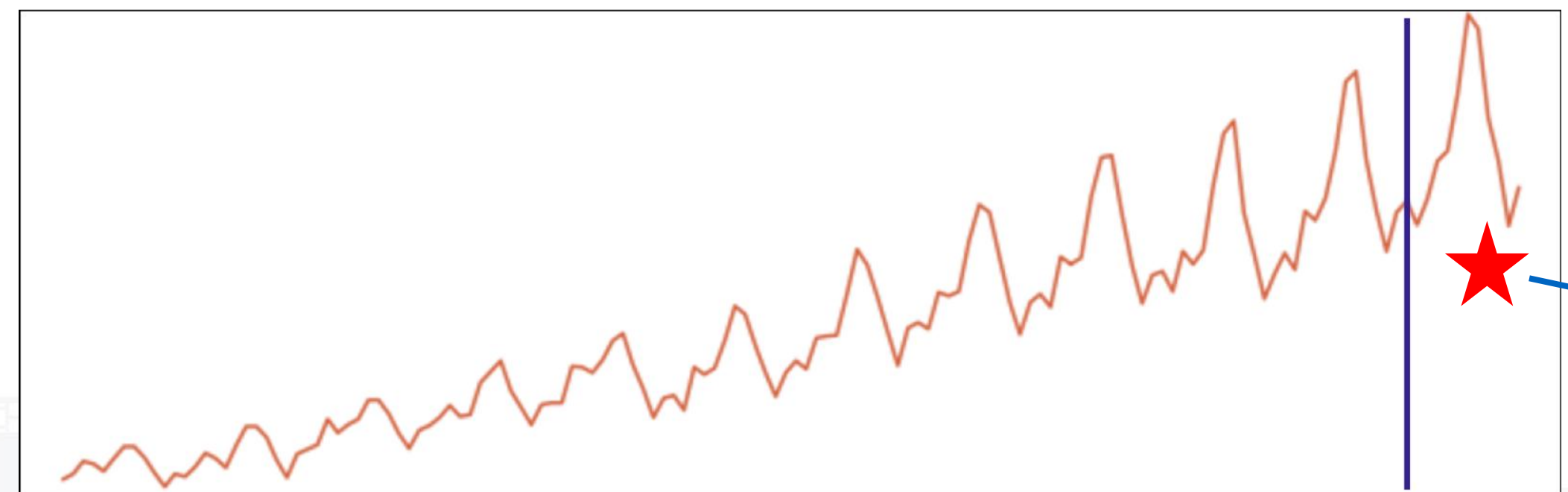
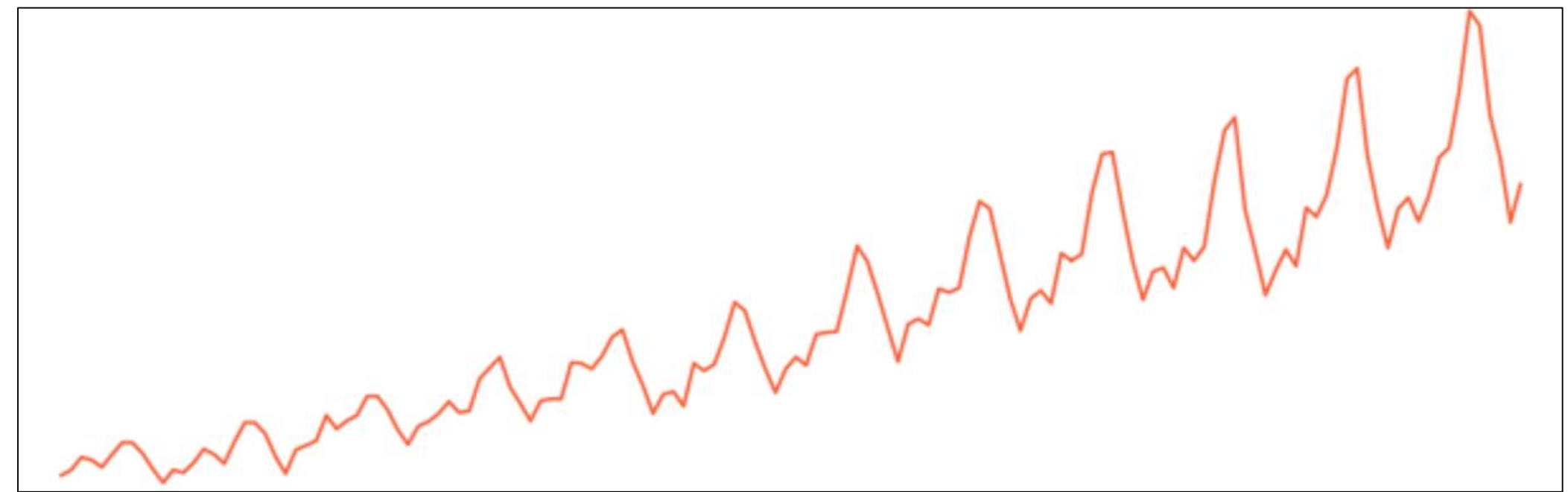
# 时间序列里的趋势

时间序列里往往包含趋势：

- 增长的趋势
- 降低的趋势

传统方法（固定阈值 / 历史平均值）

- 用过去的的数据作为标准
- 造成误判（false positive）



历史数据的  
平均值

历史数据

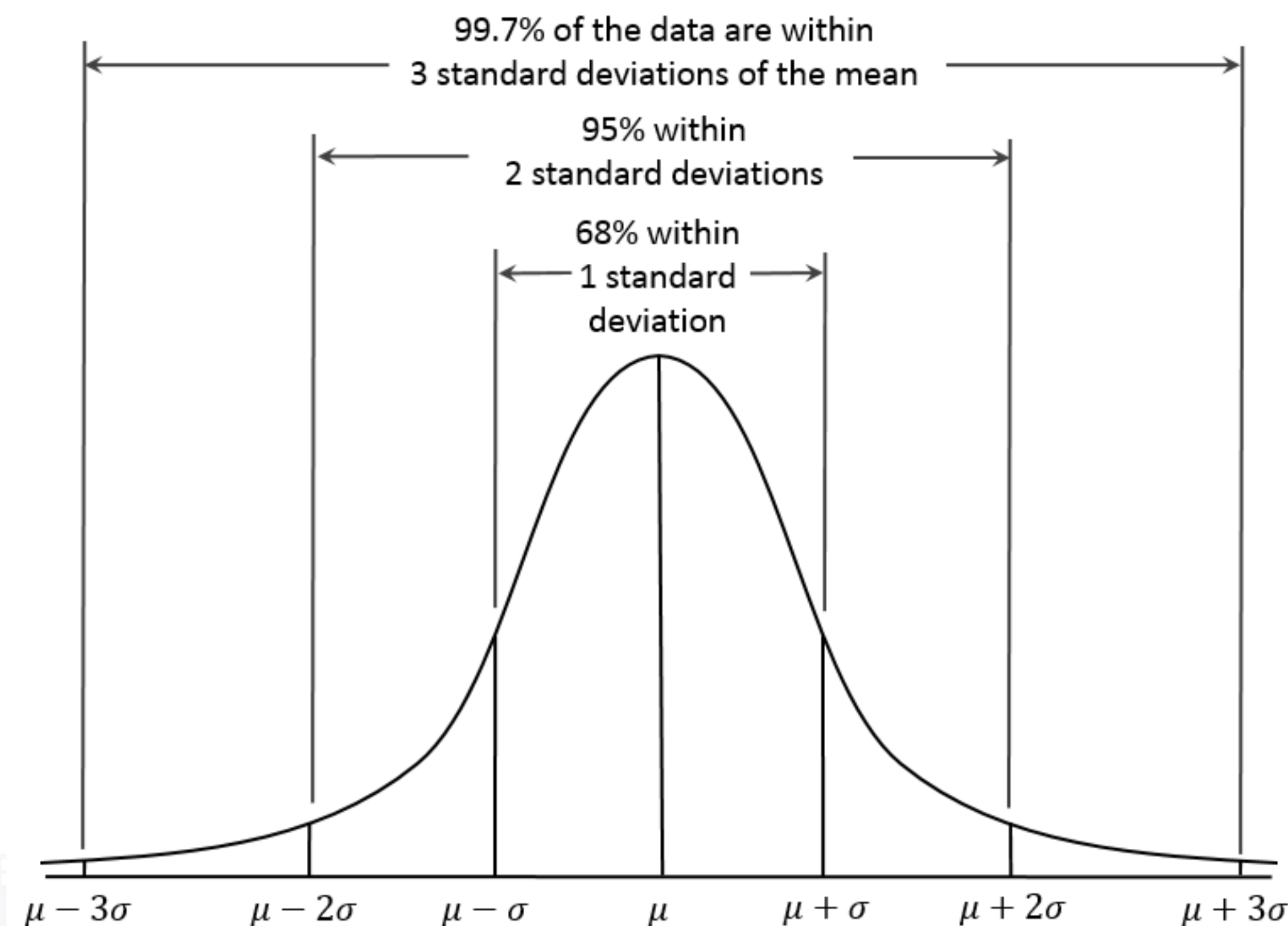
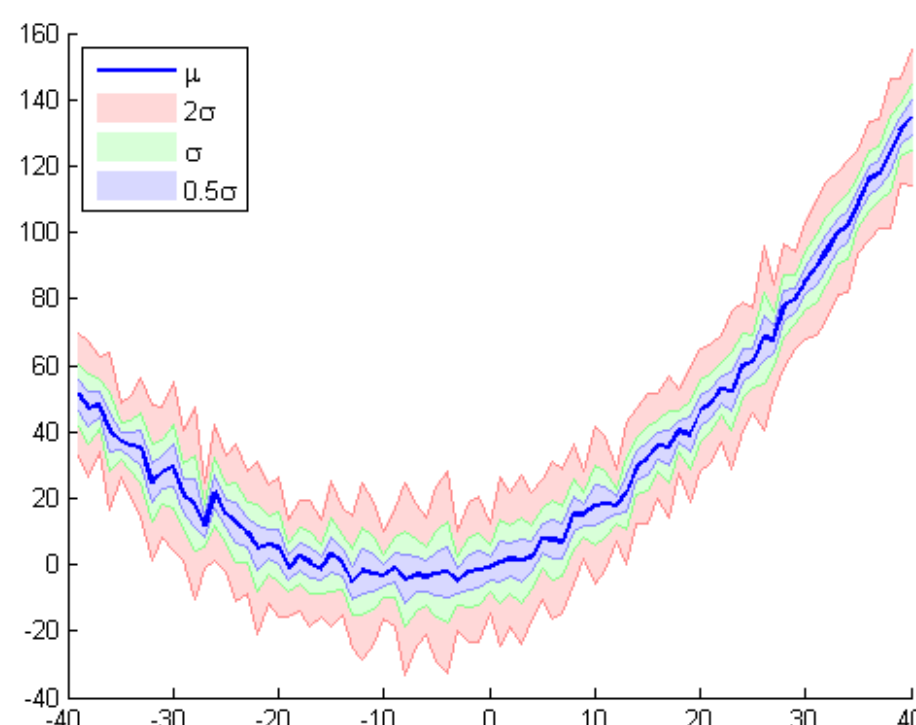
now

# 原因3: 数据是不完美的



# 理论看上去很美

- 计算
  - (1) 平均值  $\mu$
  - (2) 方差  $\sigma$
- 正常取值区间： $\mu \pm 2\sigma$
- 如果value在区间外，发出警报



基于normal distribution的假设

# 即便数据符合normal distribution...

假设每个metric每一分钟有一个值（比如errors per min, avg latency per min等）

- 每天有： $1 \times 60 \times 24 = 1,440$  数据点
- 假设我们用 $\mu \pm 3\sigma$ 区间
- 根据normal distribution定义，99.7% 的数据在该范围内
- $1,440 \times 99.7\% = 1435.68$ 个数据点被认为正常
- 系统认为 $1,440 - 1435.68 = 4.32$ 个数据点为异常

这意味着：

- 即使完全正常，每天每个metric会收到4.32个错误警报提醒（false alarms）！

# 原因4: 数据孤岛





# 数据孤岛

- 每个metric被单独考虑
- 系统里有成千上万个metric
- 彼此相互联系 / 关联
- 能否把所有的metrics联合起来一起考虑？

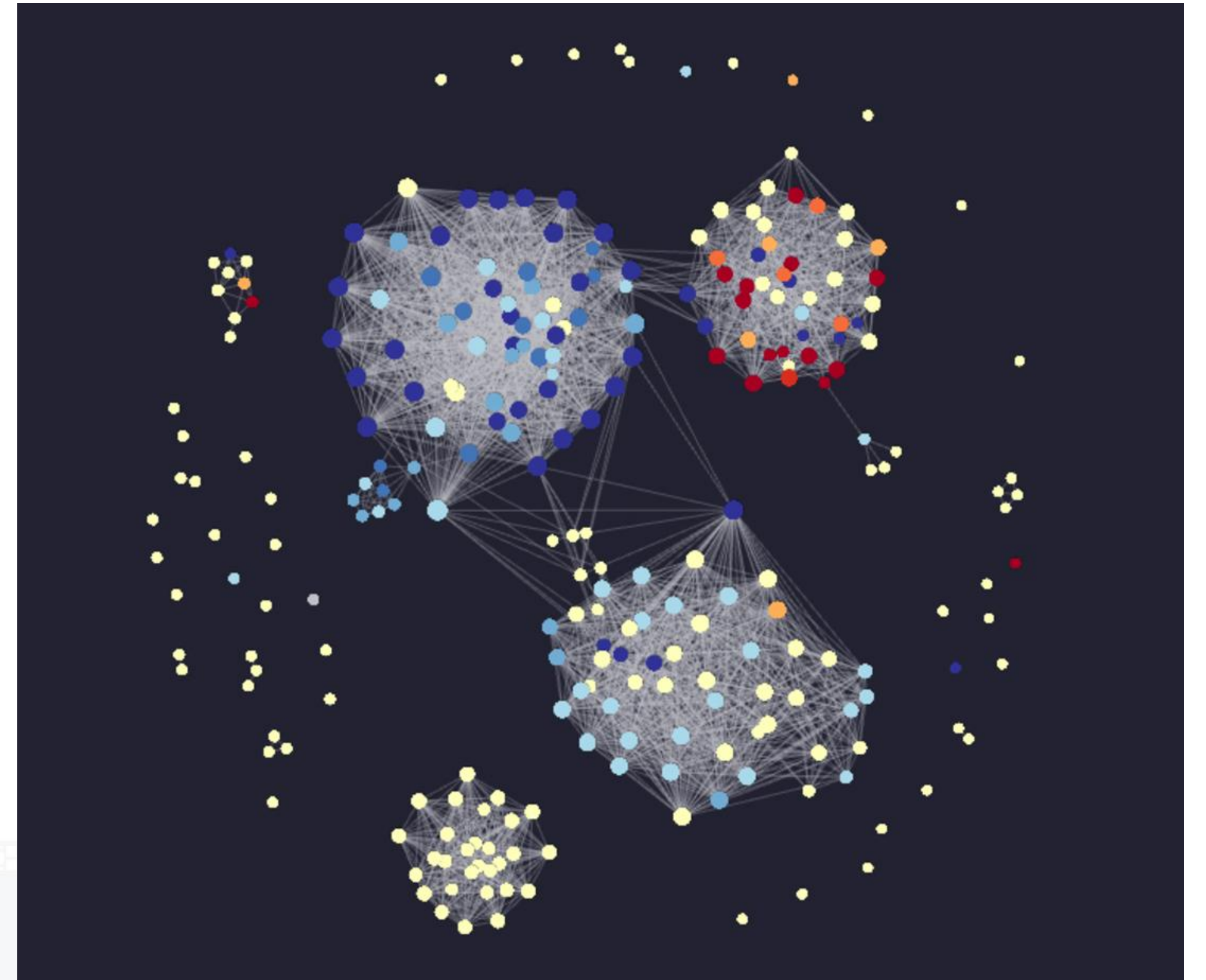


Photo Credit: Argos @ Uber

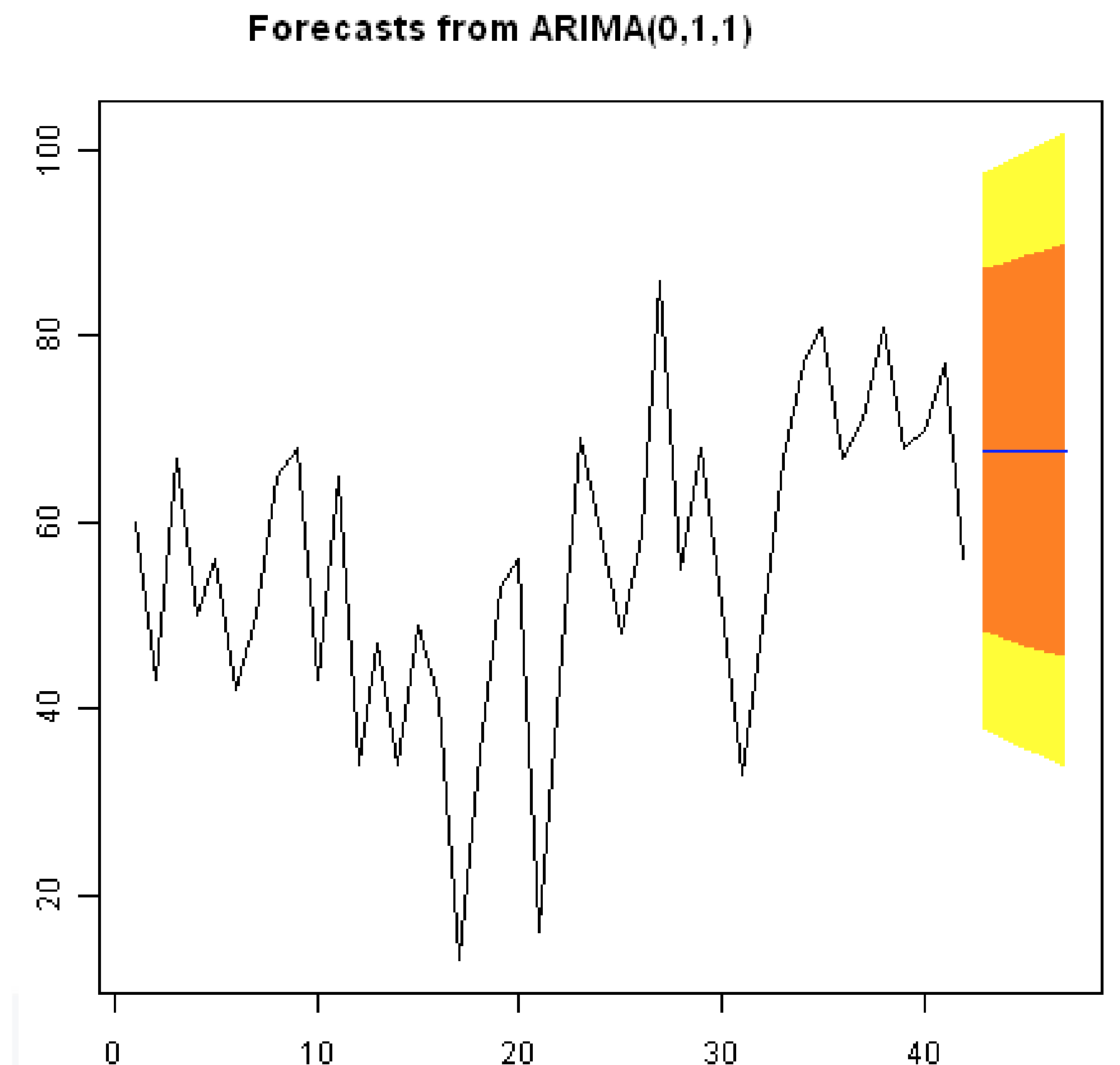
# 智能运维里的时间序列

- 性能监控里的时间序列
- 传统方法
- AI + 时间序列
  - 异常检测
  - 根源分析
- 总结



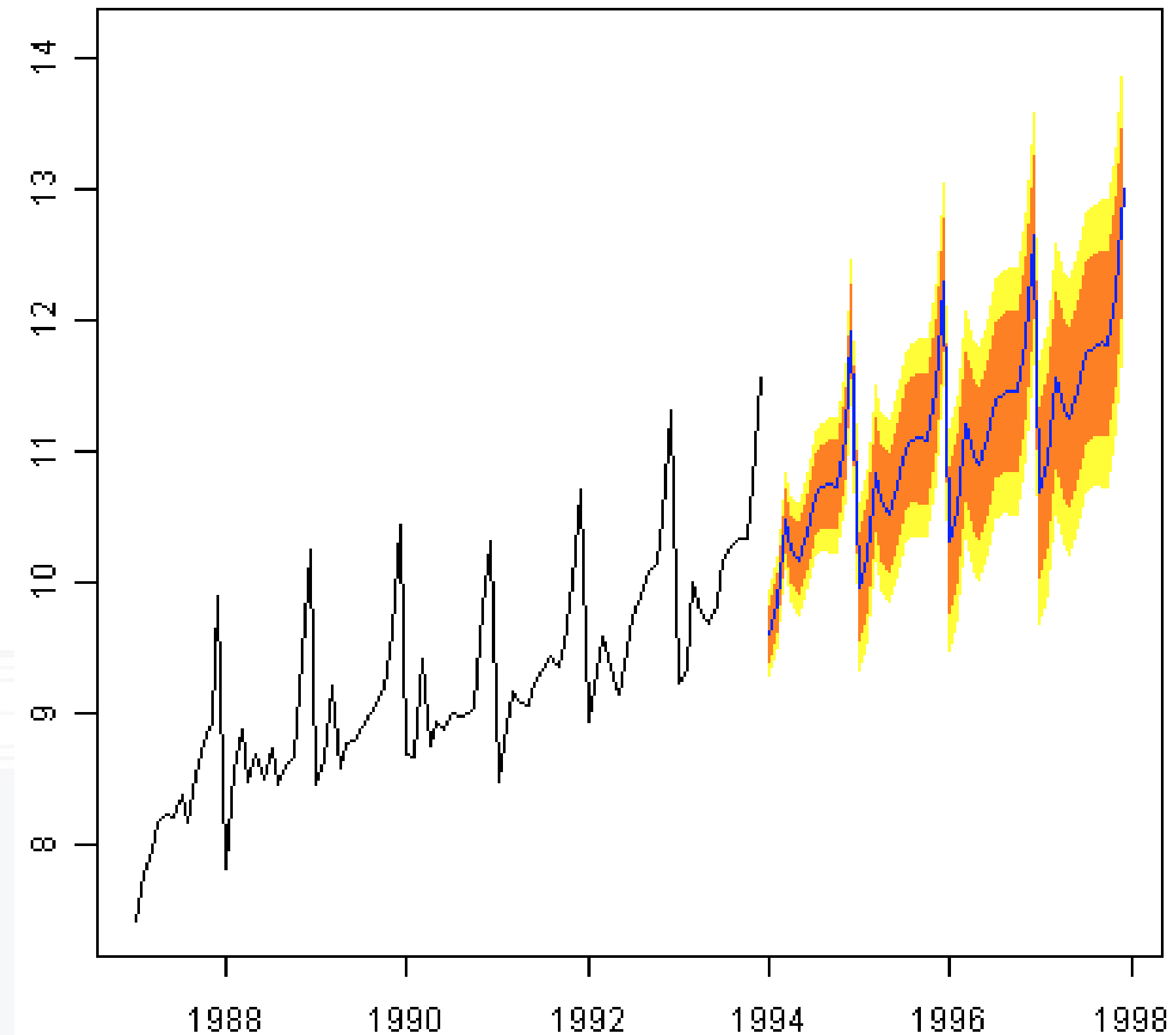
# Moving Average Based

- 对历史值赋予不同权重
- Autoregressive Integrated Moving Average (ARIMA)
- 几乎所有的time series包里都有实现
- 缺点：
  - 需手动设置参数
  - assumption较为简单



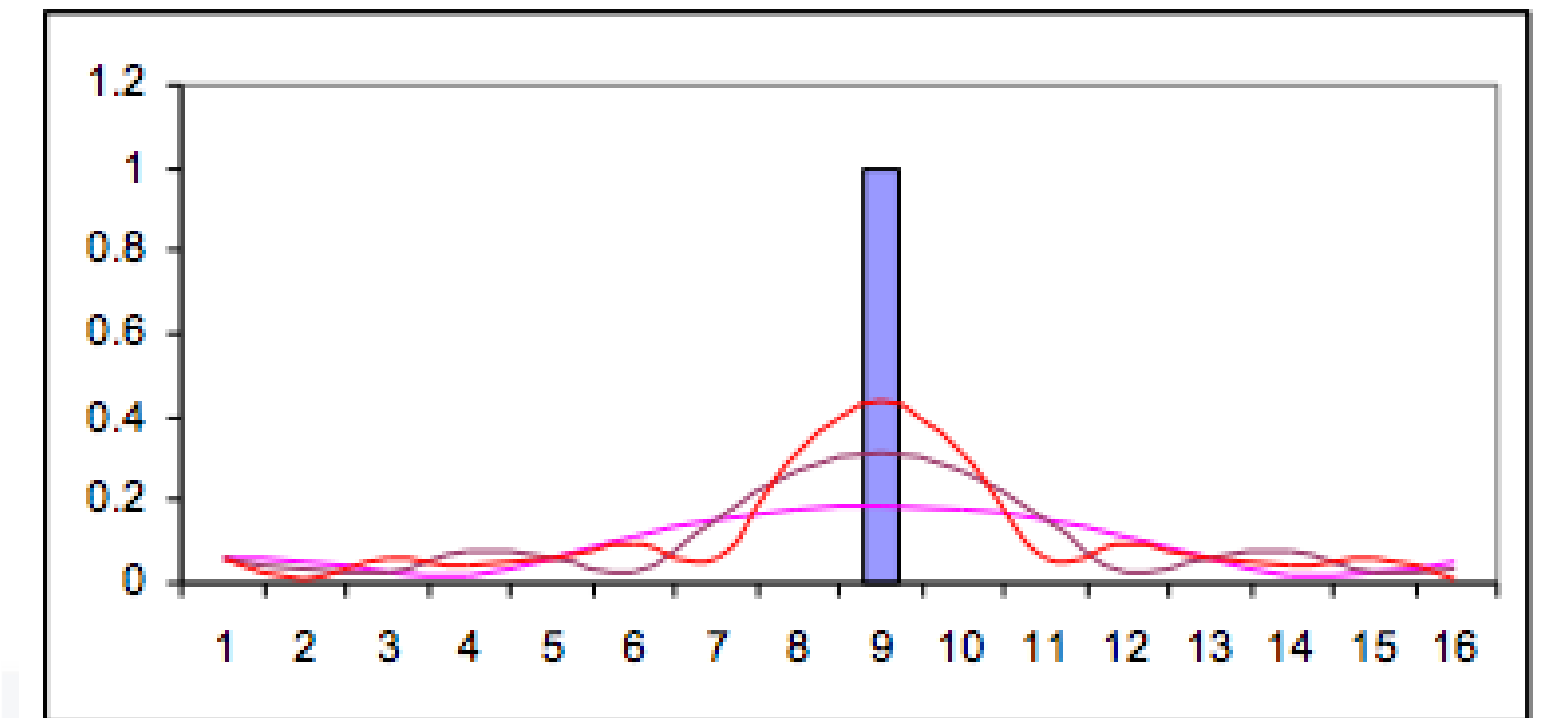
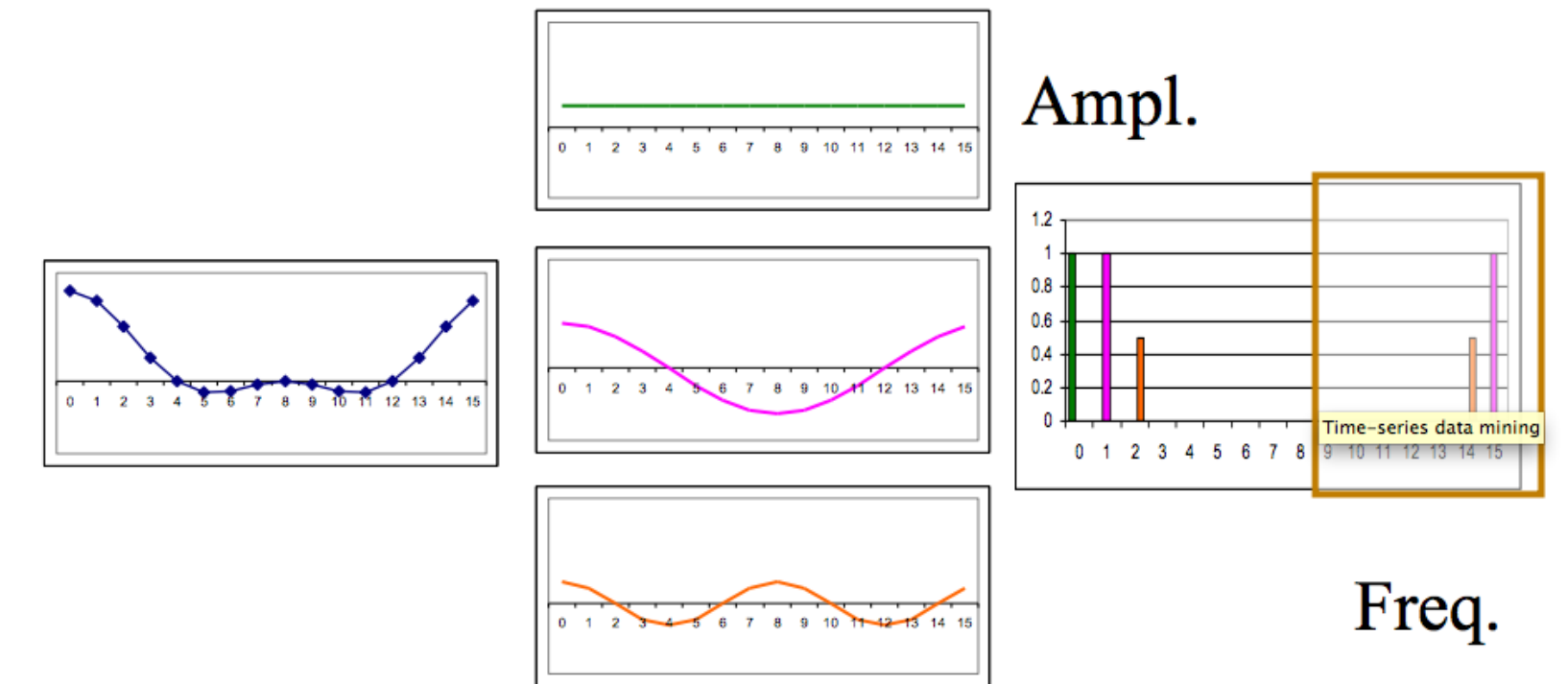
# Exponential Smoothing Based

- Double exponential (Holt-Linear):
  - 能追踪level和trend
- Triple exponential:
  - 能追踪level和trend
  - 还能追踪seasonality
- 缺点：
  - 需手动设置参数



# 分解 ( decomposition ) Based

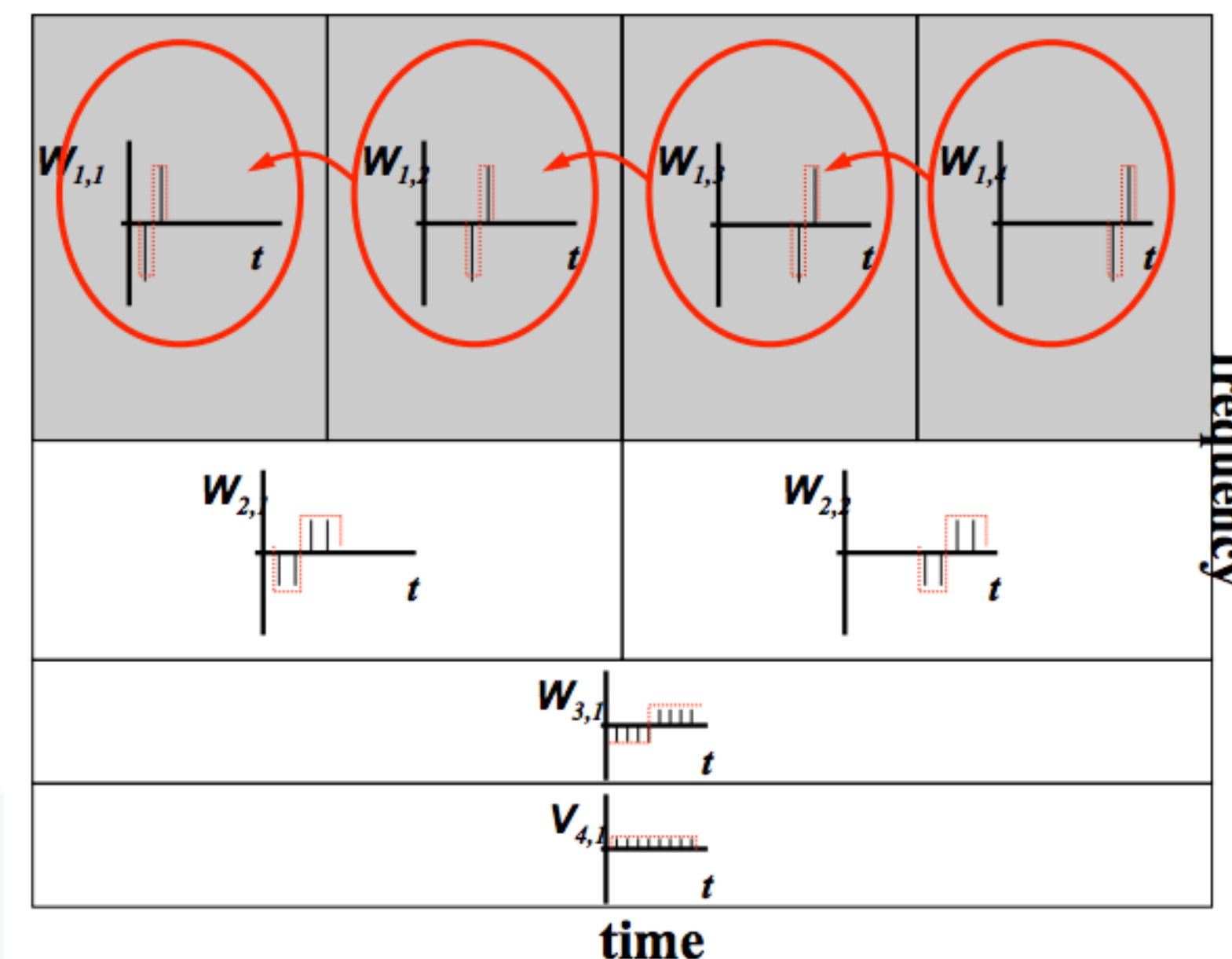
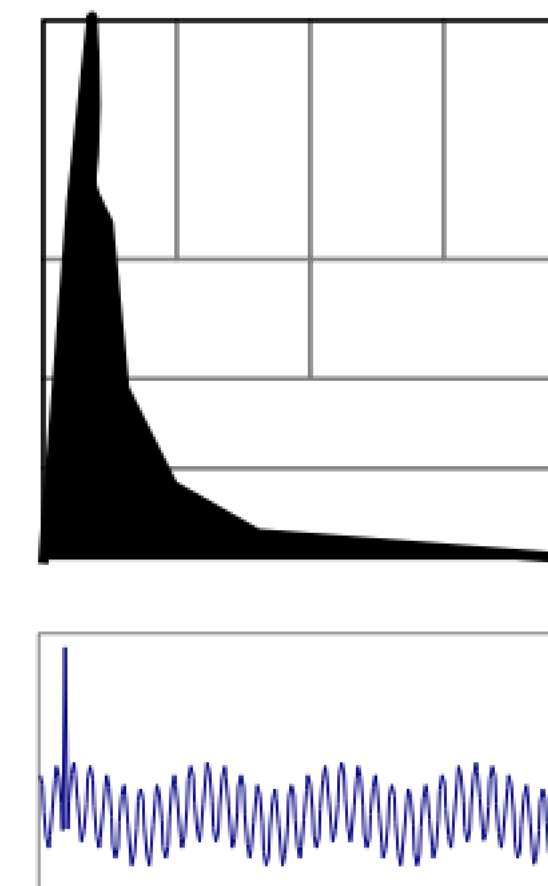
- DFT (Discrete Fourier Transform) / 傅立叶分解
  - 对周期性数据效果较好
  - 对spike model能力较弱





# 分解 ( decomposition ) Based

- DWT (Discrete Wavelet Transform) / 小波分解
  - 速度快，压缩能力强
  - 对spike等异常非常有效
- 在分解后的模块（比如小波分解后的components）上再进行auto regression

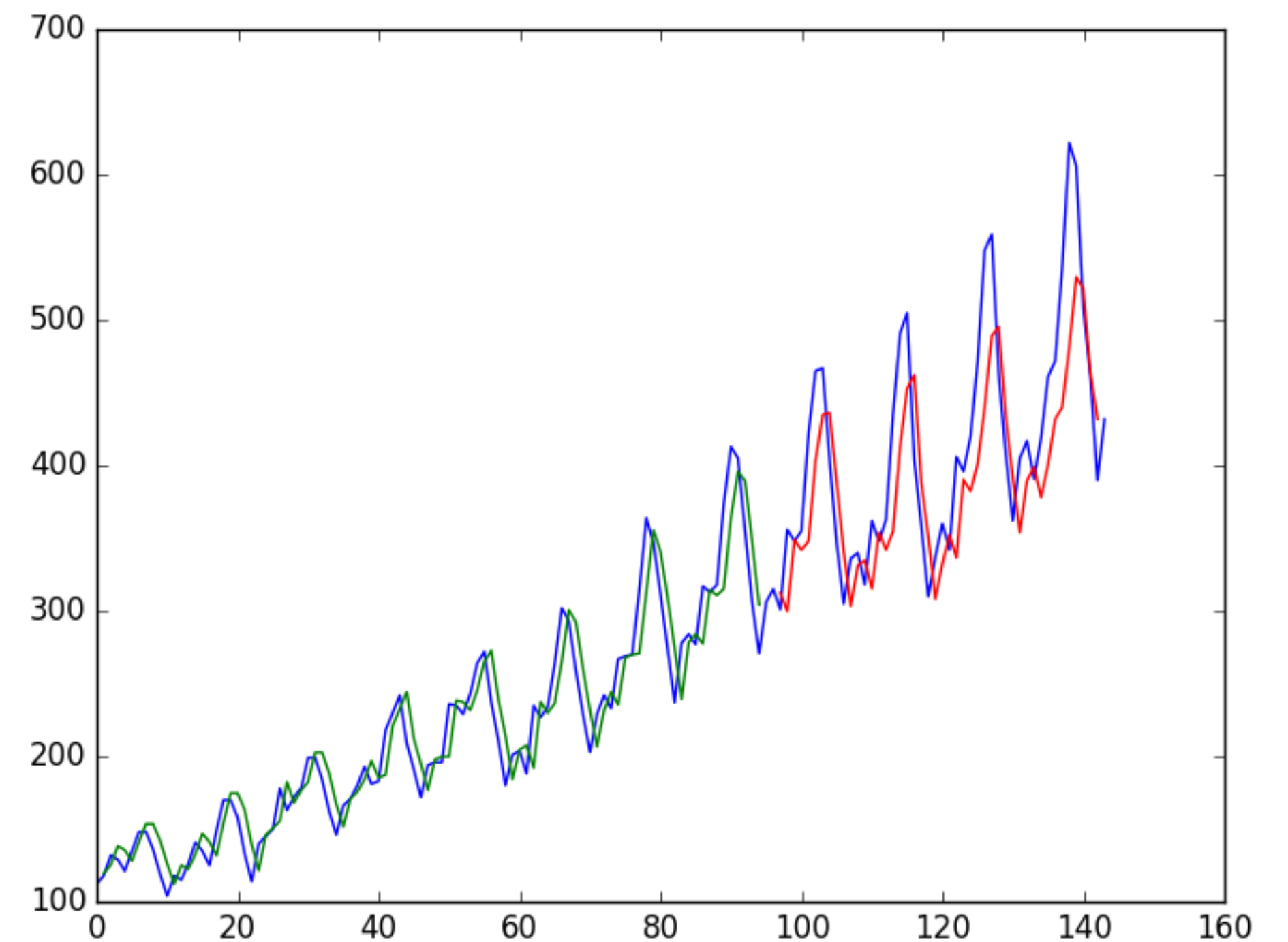


# 深度学习 ( Deep Learning ) Based

- Feedforward Neural Network
- Recurrent Neural Network (e.g., LSTM)
- Convolutional Neural Network

LSTM 例子:

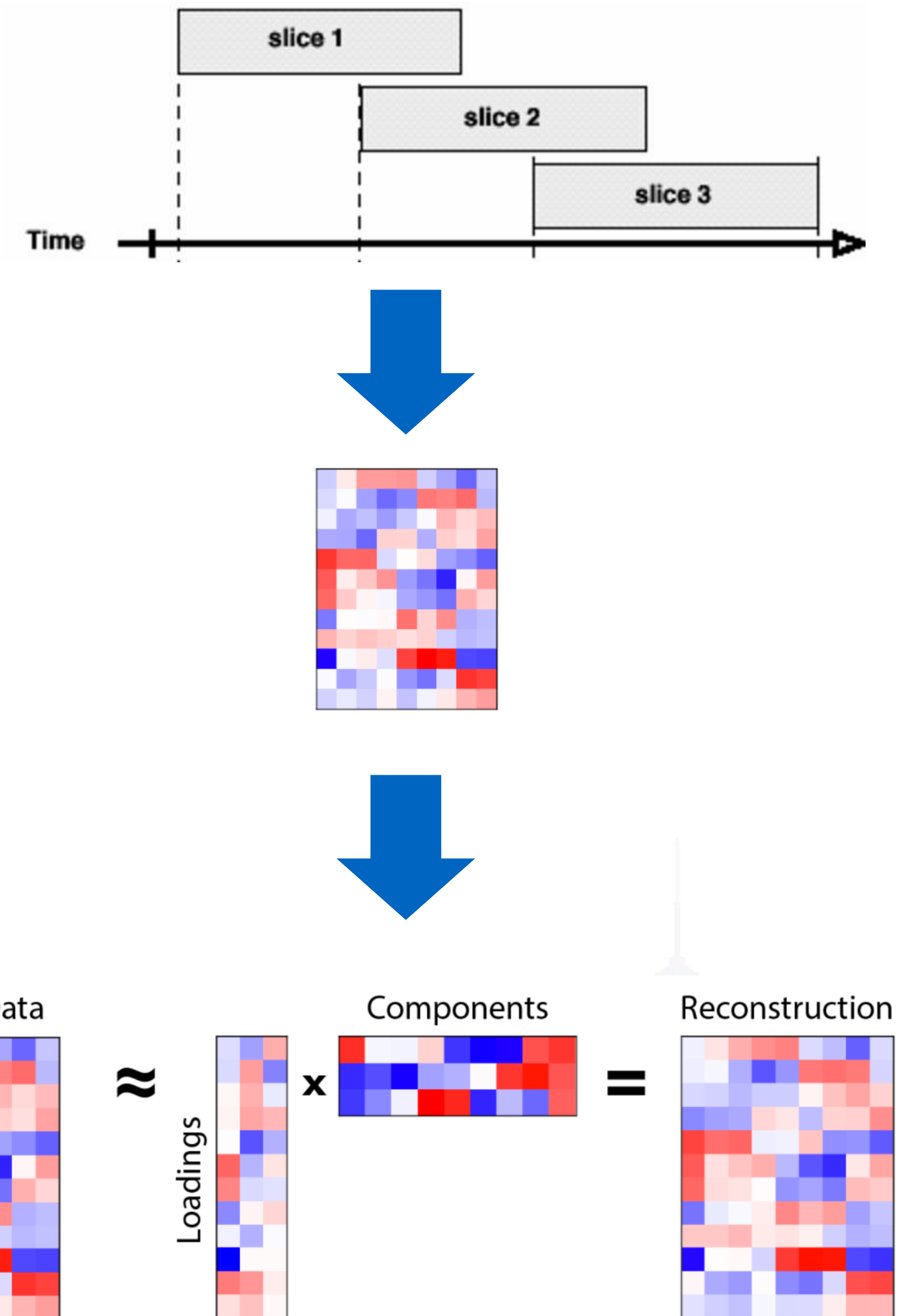
- 1 input
- a hidden layer with 4 LSTM neurons
- an output layer



# 矩阵 ( Matrix ) Based

用sliding window构建矩阵，假设矩阵为low rank  
分解矩阵后重建，差异大的地方为异常

- Principle Component Analysis
- SVD
- Robust PCA
- Auto-Encoder Neural Network
- Convolutional Auto-Encoder Neural Network



# 业界应用

- Twitter: **Seasonal Hybrid ESD (S-H-ESD)**

<https://blog.twitter.com/2015/introducing-practical-and-robust-anomaly-detection-in-a-time-series>

- Netflix: **Robust PCA**

<http://techblog.netflix.com/2015/02/rad-outlier-detection-on-big-data.html>

- Numenta: **neural network**

<http://numenta.com/press/2015/11/10/numenta-anomaly-benchmark-nab-evaluates-anomaly-detection-techniques/>

- Anodot: **online machine learning algorithm**

<http://www.anodot.com/>

- LinkedIn: **exponential smoothing**

<https://github.com/linkedin/luminol>

- Uber: **multivariate non-linear model**

<https://eng.uber.com/argos/>

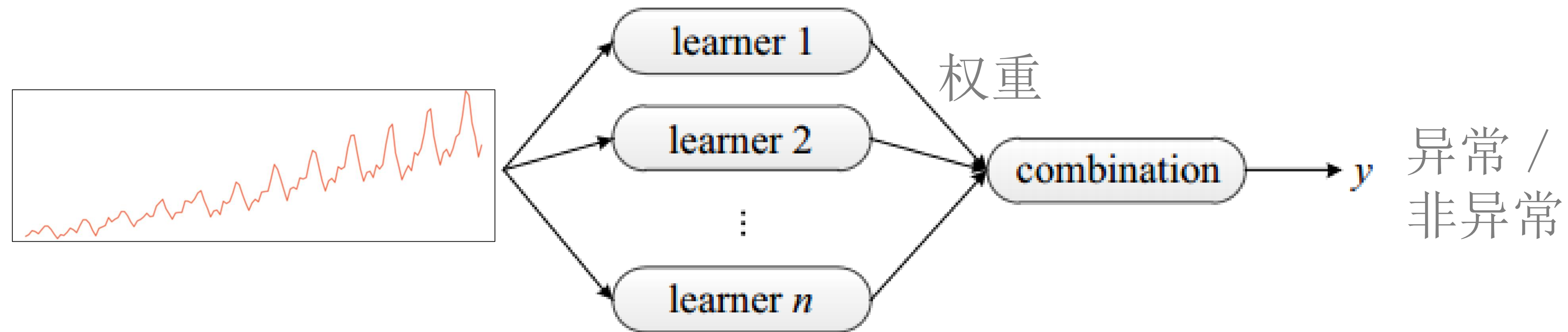
Credit: Yi Hong @ AppDynamics

# 这么多方法，选哪个呢？



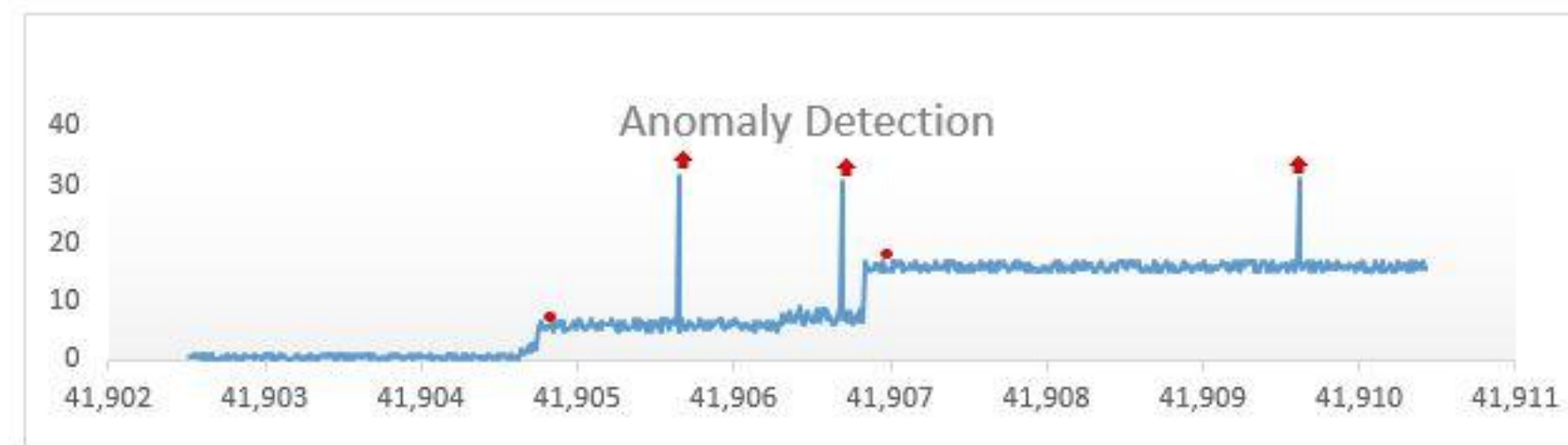


# 自动模型选取：Ensemble Learning



- 多个模型同时预测
- 根据历史数据调整不同模型权重
- 无需人工选取 / 调整
- 自动得到一个共同决策

- 异常检测
  - 产生警报



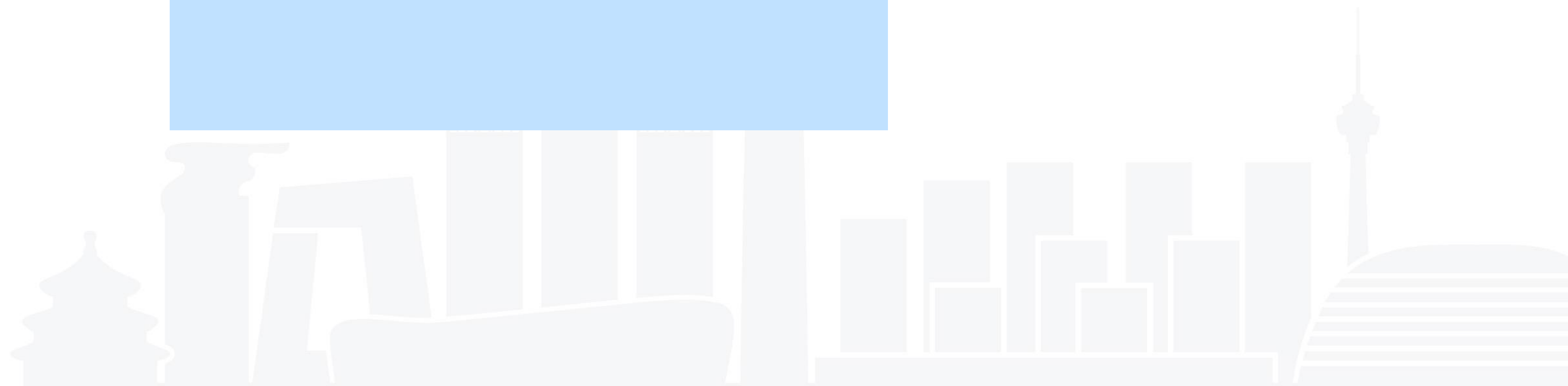
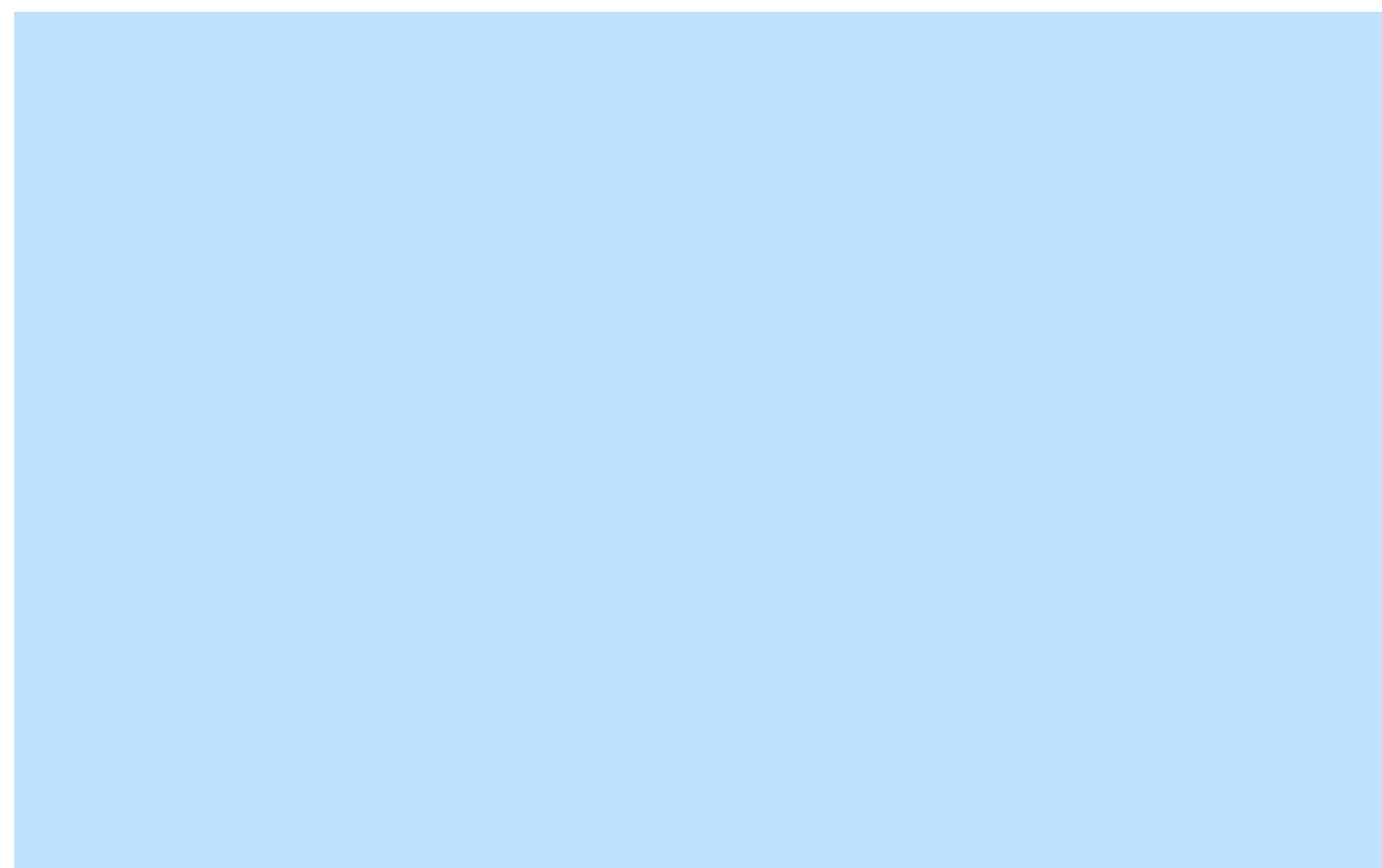
**WHAT'S**  
**NEXT STEP** your **?**

# 智能运维里的时间序列

- 性能监控里的时间序列
- 传统方法
- AI + 时间序列
  - 异常检测
  - **根源分析**
- 总结

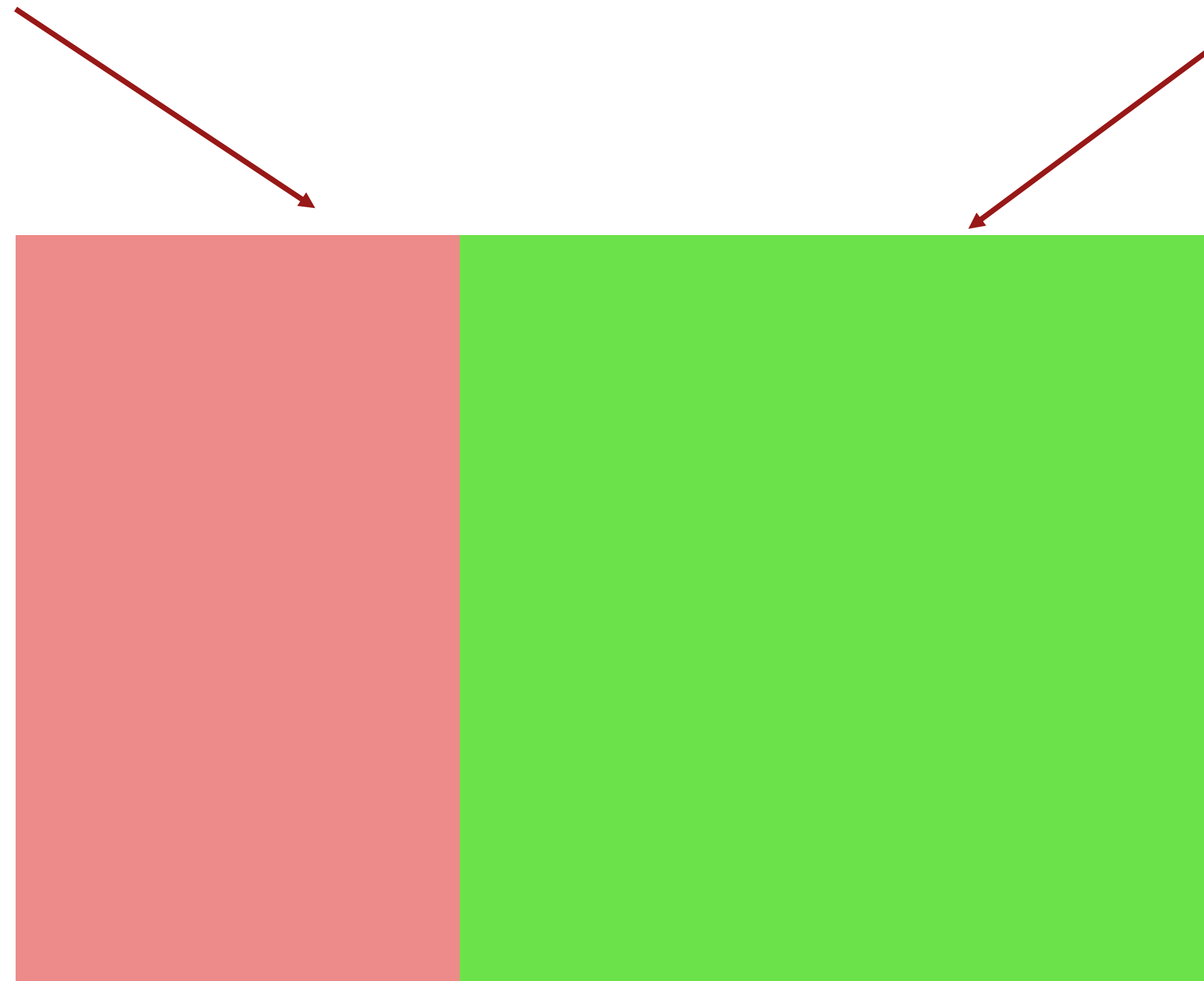


# 所有metrics集合



# 异常的metrics

# 正常的metrics



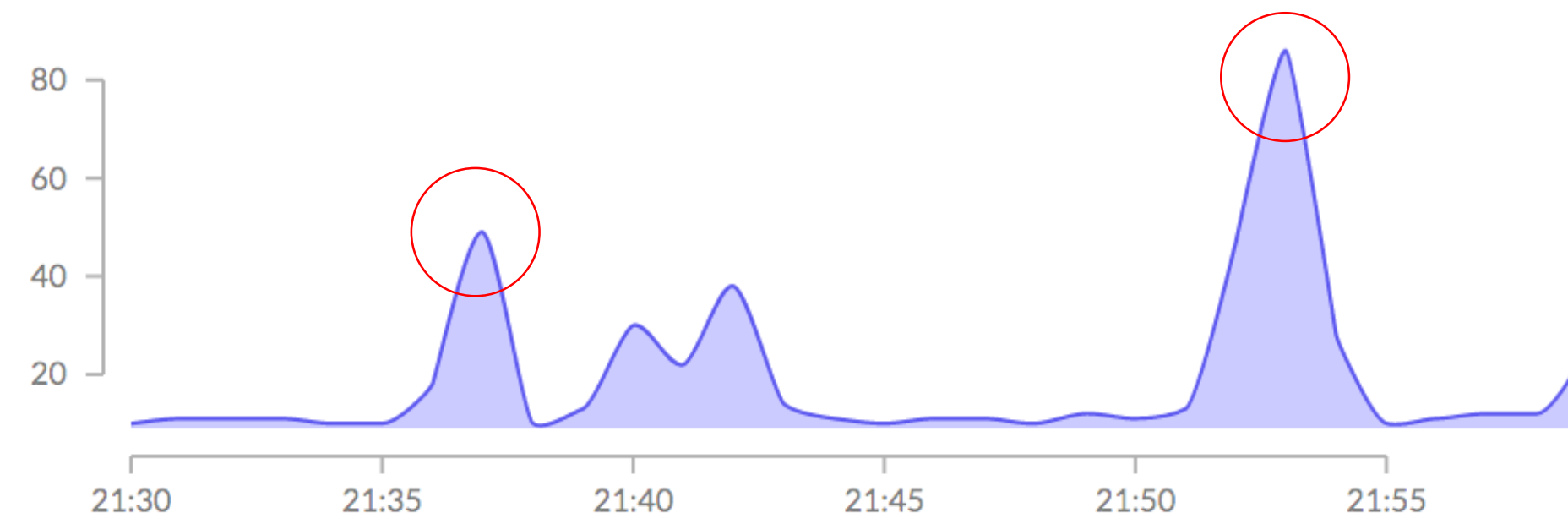


# 基于相关性的RCA

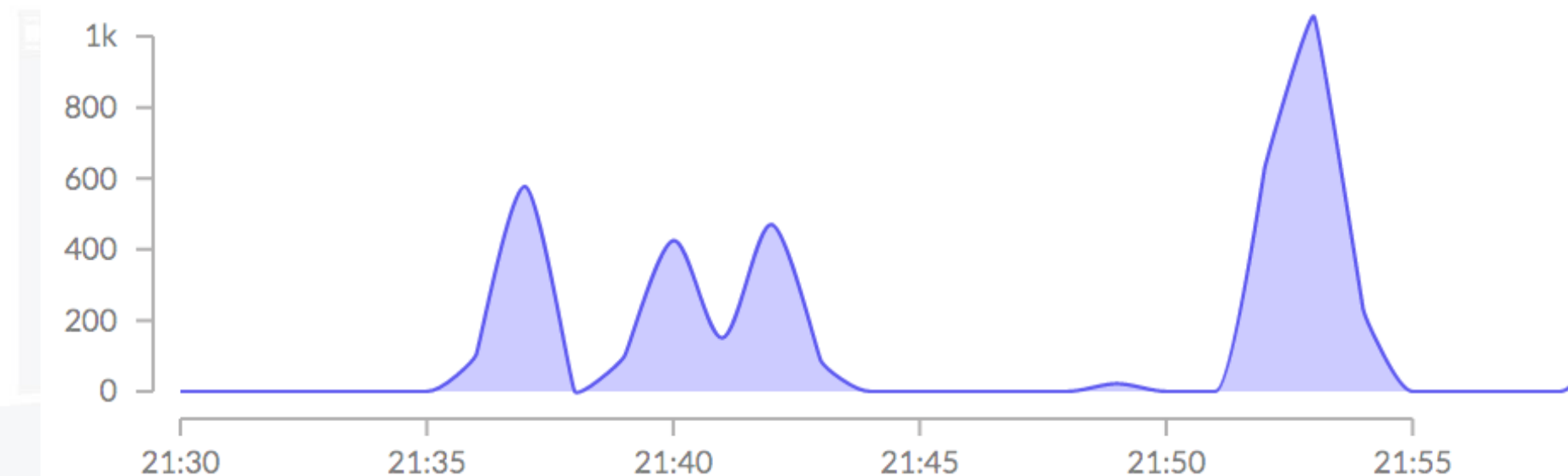
异常的metrics

- 从异常的metrics里找到最相关的metric
- 算法：Pearson correlation
- 生产环境例子：

警报metric：  
Overall Response Time



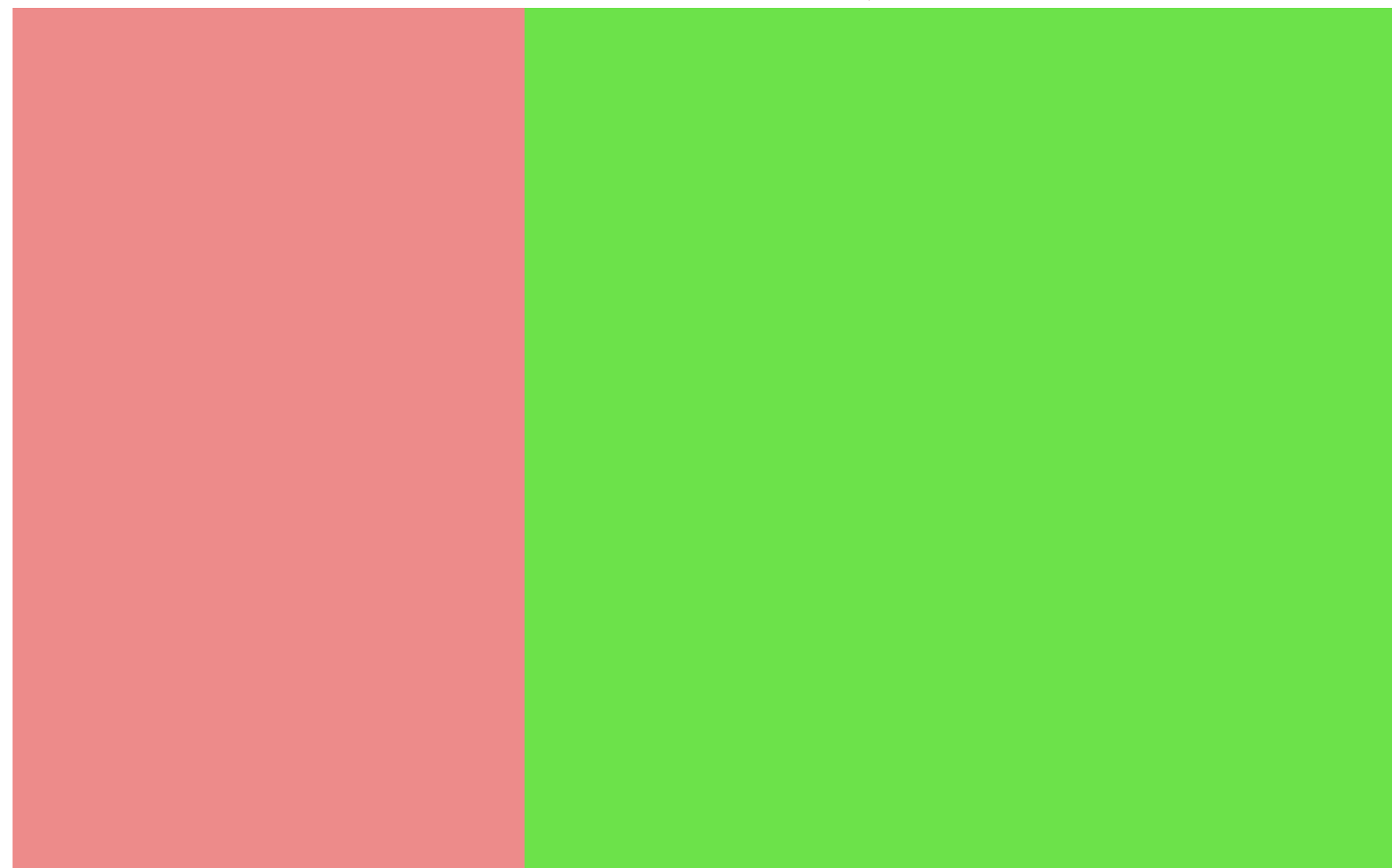
最相关的metric ( 相关度0.9876) :  
**es\_data-02**|QueryPhase|Slow Calls



# 基于决策树的RCA

异常的metrics

正常的metrics



- 将异常 / 正常metrics分成labeled data
- 算法：decision tree ( 或其他classifier )
- 总结出异常metrics的规律
- 生产环境例子：

生成的Rules：

- EsIndexCluster=**prd28**-7 && node=**indexer\_insert**\_001\_**prd28**
- Application=**prd28**-analytics && EsIndexCluster=**prd28**-2 && transactionName=**Insert**EventIndexingStage
- EsIndexCluster=**prd28**-1 && tier=**indexer\_prd28**



RCA：

- ES Cluster = prd28, stage = insert, tier = indexer

# AI + 性能监控

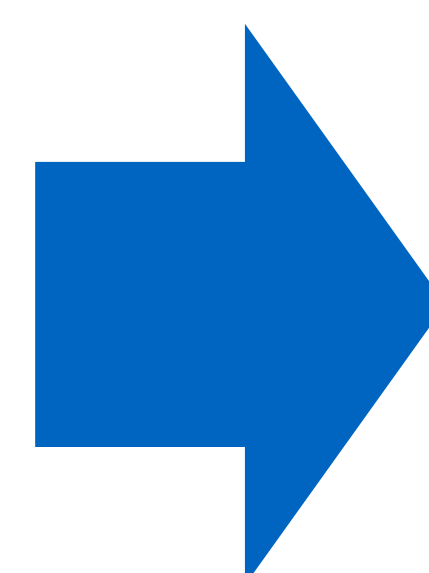
异常检测

被动式  
(reactive)



根源分析

实时  
(real time)



预测

前瞻性  
(proactive)



# 智能运维里的时间序列 - 总结

- 性能监控里的时间序列
- 传统方法
  - 固定阈值
  - 基于平均值的动态阈值
- 为什么传统方法不work？
  - 忽视周期性
  - 忽视趋势
  - 数据不完美
  - 数据孤岛
- AI + 时间序列
  - 异常检测
    - Moving average based
    - Exponential Smoothing Based
    - 分解（decomposition）Based
    - 深度学习（Deep Learning）Based
    - 矩阵（Matrix）Based
    - 自动模型选取：Ensemble Learning
  - 根源分析
    - 基于相关性的RCA
    - 基于决策树的RCA
  - 预测





关注QCon微信公众号，  
获得更多干货！

