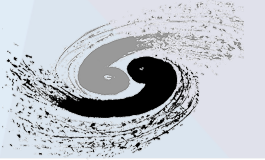
# 面向存储系统的异常行为智能检测系统

侯思琦 程垚松

中国科学院高能物理研究所计算中心

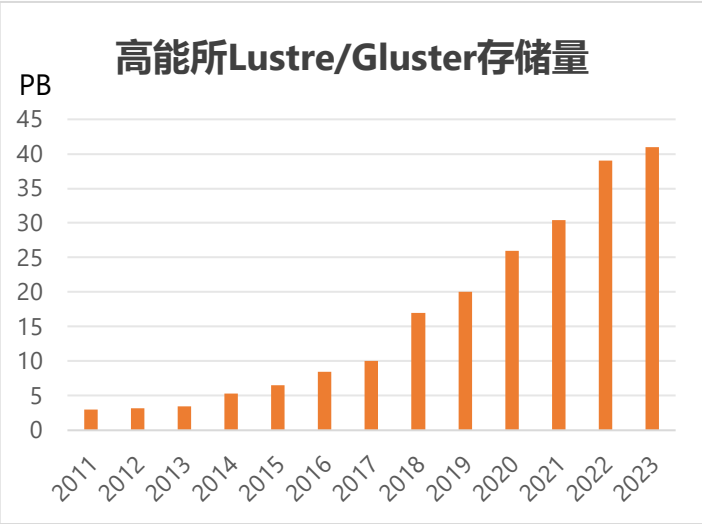
2025年8月25日 Monday

# Outline



# 背景介绍

- 大规模存储系统在数据时代的重要性
  - 在HPC、HTC领域，数据是基础底座
  - 科学计算、人工智能等数据密集型应用的数据特点
    - 数据**增量大**、数据**访问频率高**、**用户数量多**、数据**访问类型复杂多样**
  - 存储系统异常读写行为造成访问卡顿，严重影响系统正常运行
- 异常检测目标
  - 实现对异常访问行为的精准、实时检测，保障数据安全与系统稳定运行
- 核心意义
  - 提升存储系统的风险抵御能力，为数据管理提供可靠保障



# 背景介绍

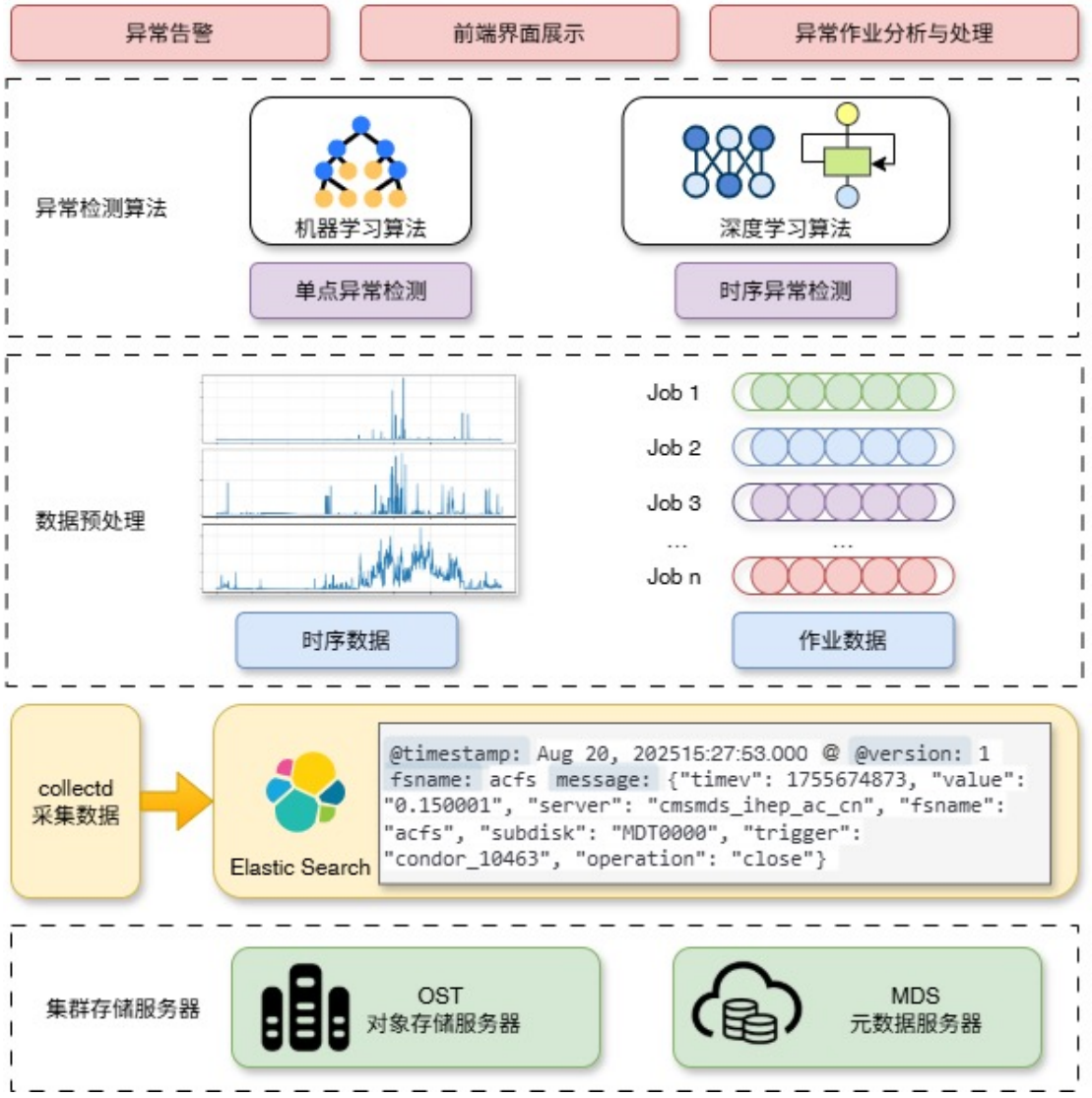
- 大规模存储系统异常检测
  - 现有方法
    - 根据机器性能指标设置静态阈值（灵活性低）
    - 通过**ping、df、touch**等命令探测（异常定位困难）
    - 用户端反馈（速度慢、影响范围大）
  - 当下挑战
    - 随着系统规模增加，静态阈值无法适配，需要实时性强的方法
    - 数据访问量频率高、访问类型复杂，人工归因定位困难

# 背景介绍

- 大规模存储系统异常特点
  - **时序性**：存储系统异常行为往往**与特定时间序列相关**
  - **稀疏性**：异常行为在存储系统的日常运行数据中**占比较低**
  - **突发性**：部分异常行为会突然发生，如用户作业出现死循环或大量文件突发读写
  - **关联性**：历史读写操作的**叠加**会对后续读写行为产生**持续影响**

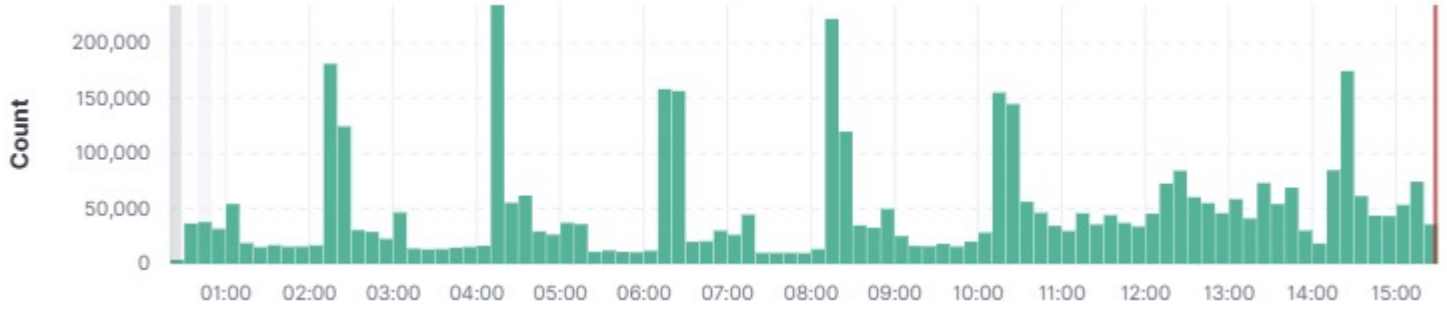
# 核心技术架构

- 面向存储系统异常行为的智能检测系统
  - 数据采集
    - 对接集群存储服务器，采集作业读写请求信息
  - 数据预处理
    - 数据清洗、标准化，提取有效特征
  - 智能检测算法
    - 实施异常检测算法，机器学习、深度学习等
    - 算法优化与迭代训练
  - 决策与响应
    - 根据检测结果执行响应，如告警、配置调整等



# 核心技术架构

- 数据采集
  - 采集对象
    - 元数据服务器上的元数据目标（**MDT**）、对象存储服务器上的对象存储目标（**OST**）




  - 采集工具：**collectd plugin**
  - 信息存储：**Elasticsearch**
  - 数据记录针对**MDT**或**OST**的读写请求，包括数据操作、对应命令、**uid**、时间戳等

| Time | Document |
| --- | --- |
| Aug 22, 2025 @ 16:12:21.000 | fsname: hepsfs @timestamp: Aug 22, 2025 @ 16:12:21.000 @version: 1 message: {"timev": 1755850341, "value": "0.483334", "server": "hepsmds01_ihep_ac_cn", "fsname": "hepsfs", "subdisk": "MDT0000", "trigger": "df_60043", "operation": "statfs", "type": "lustremdtcollectdmsg"} operation: statfs server: hepsmds01_ihep_ac_cn subdisk: MDT0000 timev: 1,755,850,341 trigger: df_60043 type: lustremdtcollectdmsg value: 0.483 _id: 1c7W0JqBKGp1UfXwJqQZ _index: lustremdtcollectdmsg- |

# 核心技术架构

- 智能检测算法——分类
  - 表征方式

读写请求记录

- $\mathrm{E} = \{e_1, e_2, \cdots, e_n\}$

↓

特征表示

- $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n\}$

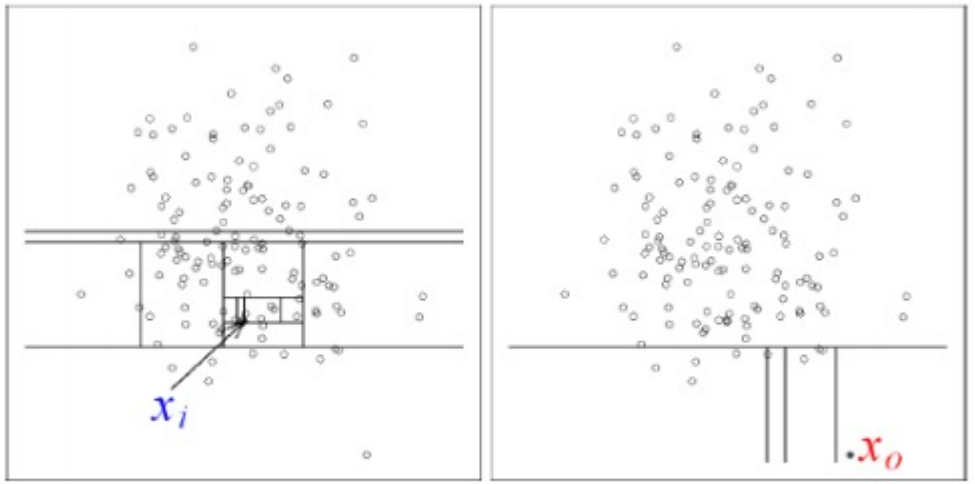
↓

异常检测

- $\mathrm{Y} = \{y_1, y_2, \cdots, y_n\}$

# 核心技术架构

- 智能检测算法——孤立森林
  - **Isolation Forest**（孤立森林），离群点检测算法
  - 假设：异常是少量且可区分的
  - 基于决策树，对数据进行划分，快速发现离群点
  - 特点
    - 无监督训练、无需标签
    - 不使用距离或密度测量，计算便捷
    - 忽略时序信息




(a) Isolating $x_i$ (b) Isolating $x_o$

# 核心技术架构

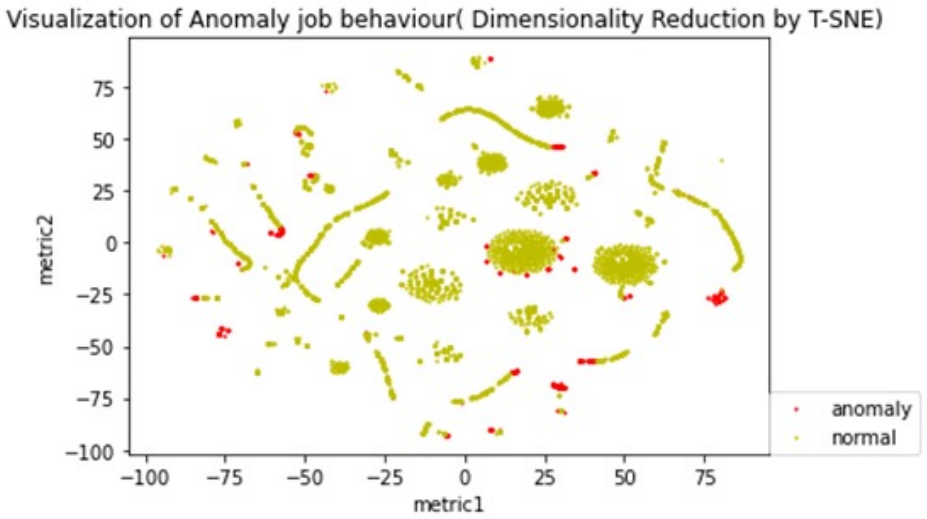
- 智能检测算法——孤立森林

**Algorithm 3 :** $PathLength(x, T, hlim, e)$

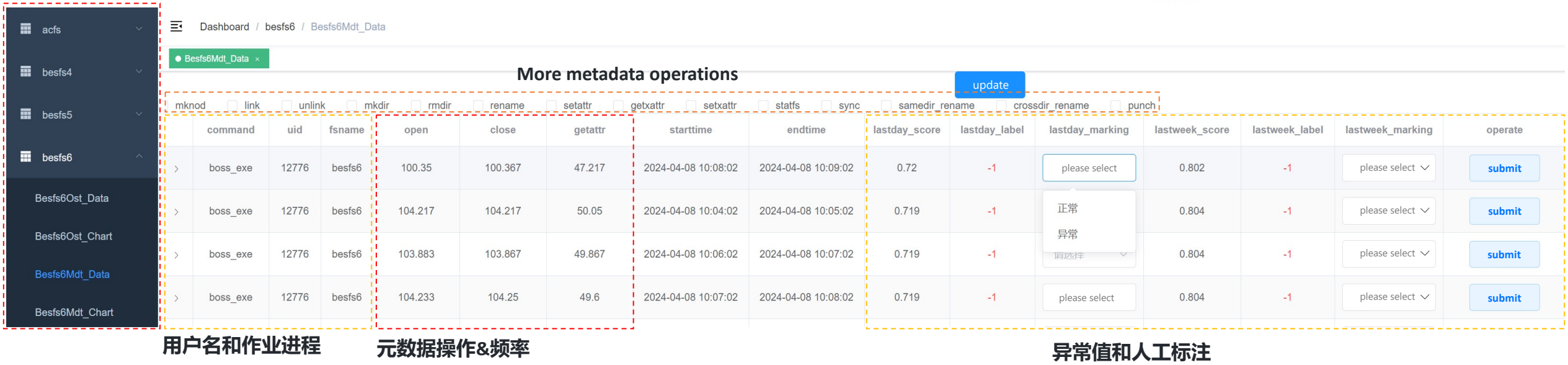
**Inputs :** $x$ - an instance, $T$ - an $i$Tree, $hlim$ - height limit, $e$ - current path length; to be initialized to zero when first called

**Output:** path length of $x$

1: if $T$ is an external node or $e \geq hlim$ then
2: return $e + c(T.size)$ {$c(.)$ is defined in Equation 1}
3: end if
4: $a \leftarrow T.splitAtt$
5: if $x_a < T.splitValue$ then
6: return $PathLength(x, T.left, hlim, e+1)$
7: else {$x_a \geq T.splitValue$}
8: return $PathLength(x, T.right, hlim, e+1)$
9: end if




**文件系统实例**

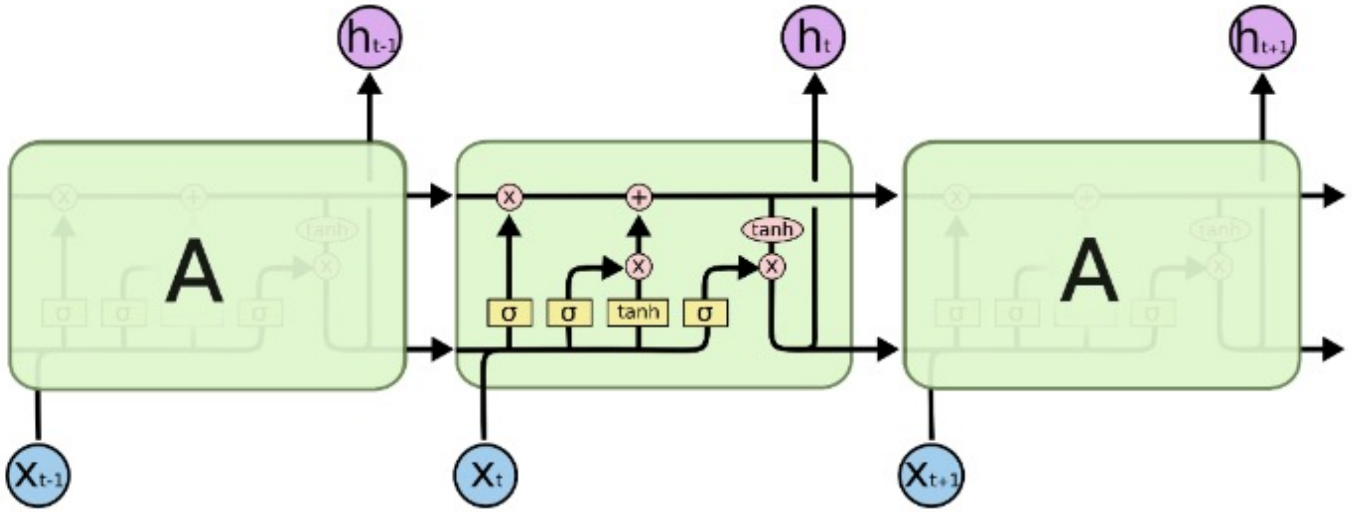



用户名和作业进程　　元数据操作&频率　　异常值和人工标注

# 核心技术架构

- 智能检测算法——LSTM
  - LSTM（长短时记忆网络），循环神经网络
  - 特征：
    - 捕捉时间序列数据中的长期依赖关系
    - 适应长周期关联，能针对存储系统中跨度较大的操作序列进行连贯学习
    - 灵活的深度学习方法

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\
g_t &= \tan h(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tan h(c_t)
\end{aligned}
$$



# 核心技术架构

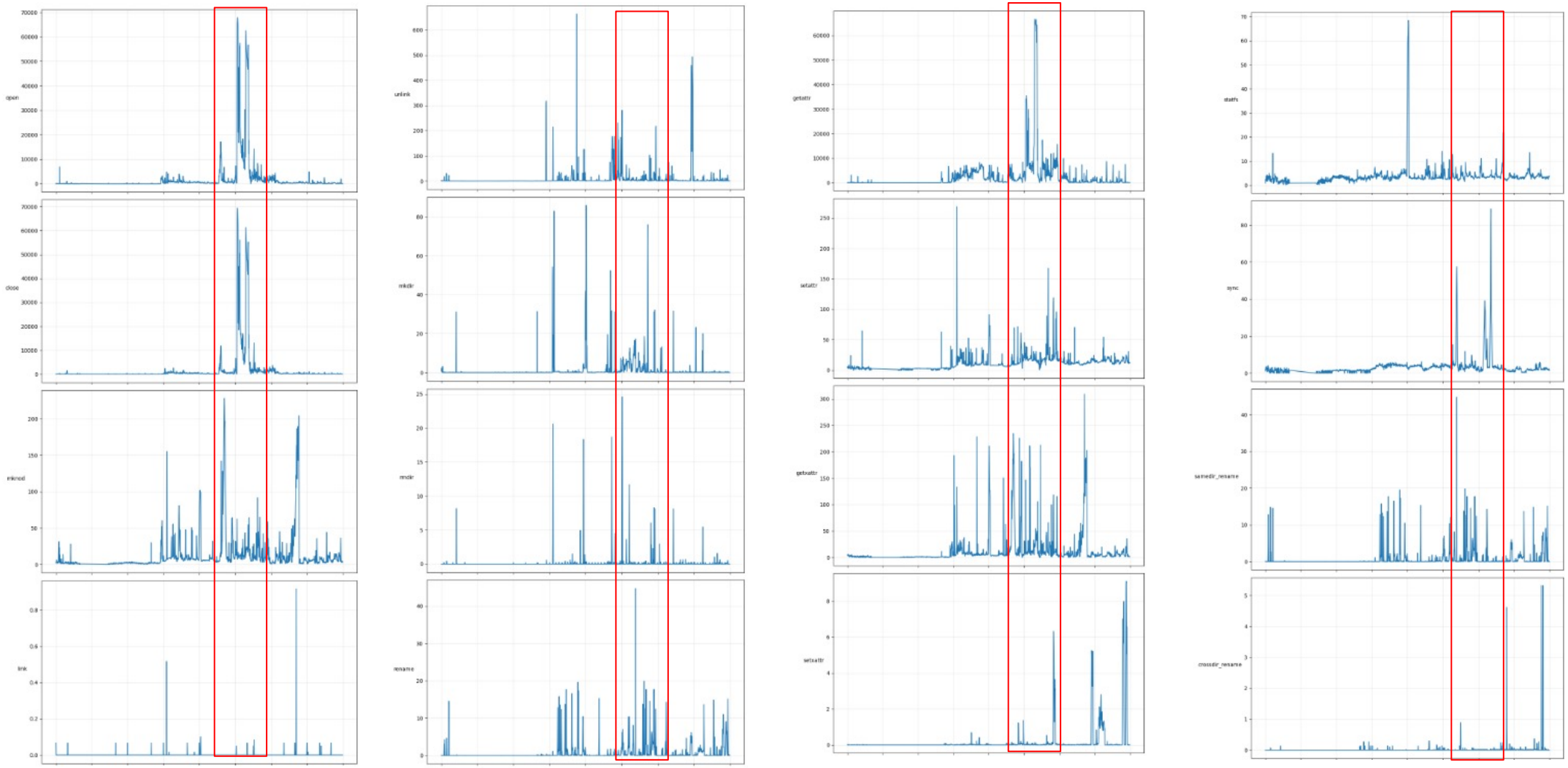
- 模型输入
  - 滑动窗口$\boldsymbol{s}_{\mathrm{i}}=\{\boldsymbol{x}_{\mathrm{i}},\boldsymbol{x}_{\mathrm{i+1}},\cdots,\boldsymbol{x}_{i+w}\}$
  - 访问记录以分钟为单位划分
  - $\boldsymbol{x}_{\mathrm{i}}$是**i**时刻的聚合特征，是一个**d**维向量，向量各个维度指示一分钟内不同元数据（对象数据）操作的频率之和

| | 特征 | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ost | read_samples | write_samples | destroy | create | get_info | set_info | quotactl | sum_read_bytes | sum_write_bytes | | | | | | | | | |
| mdt | open | close | mknod | link | unlink | mkdir | rmdir | rename | getattr | setattr | getxattr | setxattr | statfs | sync | samedir_rename | crossdir_rename | punch | |

- 数据预处理——标准化
  - 对任一维度**j**，计算均值$\mu_j=\frac{1}{n}\sum_{i=0}^{n}x_{ij}$，标准差$\sigma_j=\sqrt{\frac{1}{n}\sum_{i=0}^{n}\left|\boldsymbol{x}_{ij}-\mu_j\right|^2}$
  - 对任一$x_{ij}$，标准化后$\boldsymbol{x}'_{ij}=\frac{|x_{ij}-\mu_j|}{\sigma_j}$

# 核心技术架构

- 时序数据　MDT聚合数据示例



# 核心技术架构

- 训练阶段

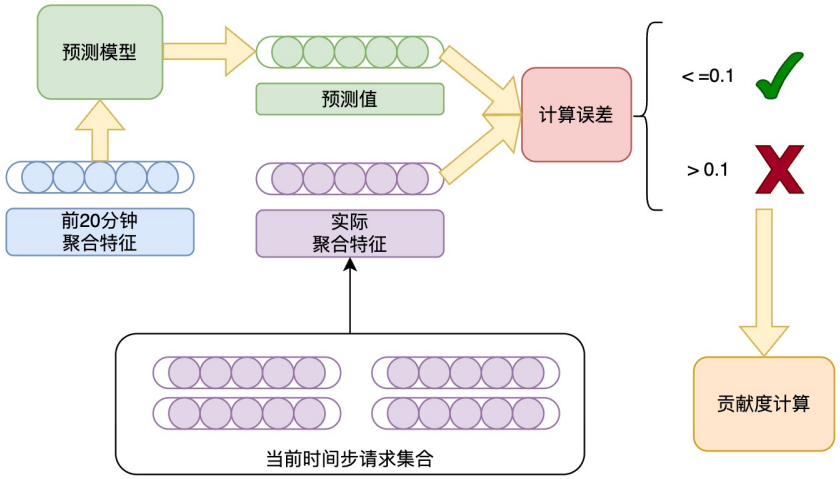
  使用预处理后的数据进行无监督训练

  训练目标：使用**LSTM**读取**20**个时间步的数据，预测下一个时间步

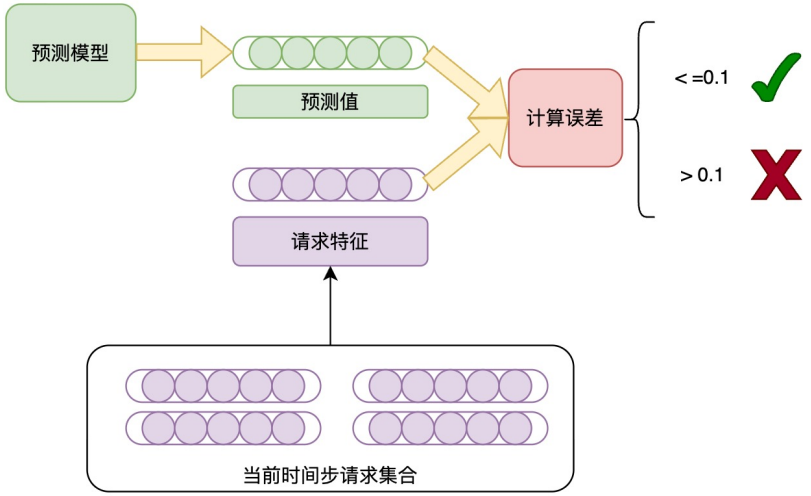
  计算预测向量与真实向量之间的均方误差作为损失，更新模型参数

$$MSE = \frac{1}{n}\|\boldsymbol{x}_{t+1} - \hat{\boldsymbol{x}}_{t+1}\|_2^2$$

- 测试阶段
  - 策略1:



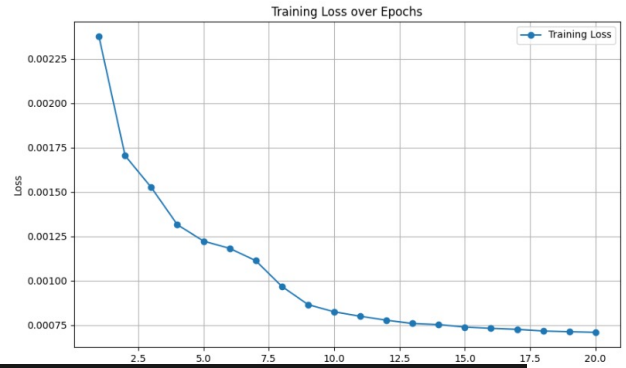

策略2:



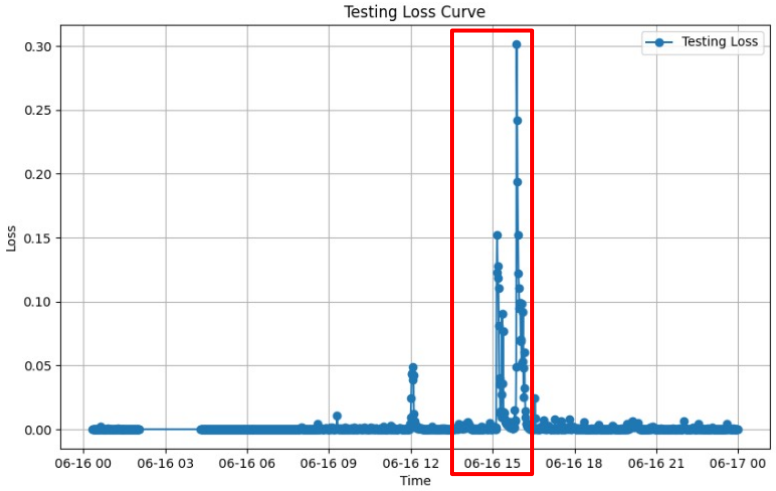
# 系统性能指标

- 检测效率
  - 使用cpu计算，实现秒级响应
- 误报率
  - 14/21，较孤立森林(29659/29669)降低33.3%

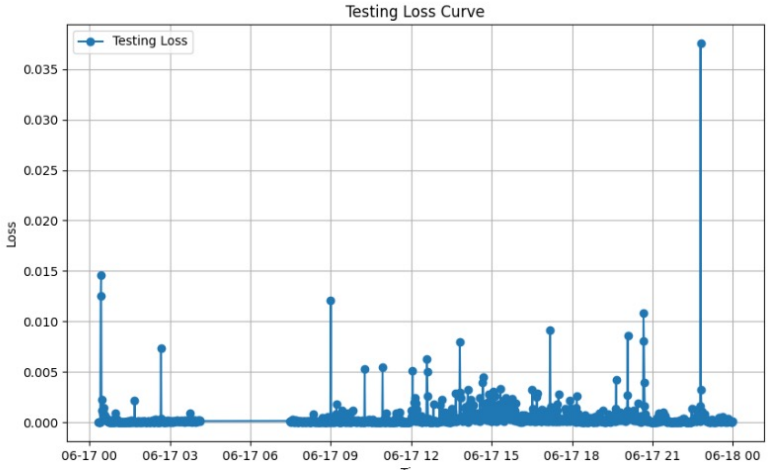
```
total number: 356085
lenth of dataset: 1286
1286it [05:09,  4.16it/s]
percentage: 0.0047741410056587 62
root_exe_22903
python_13145
17/356085 jobs are tagged as anomaly
```



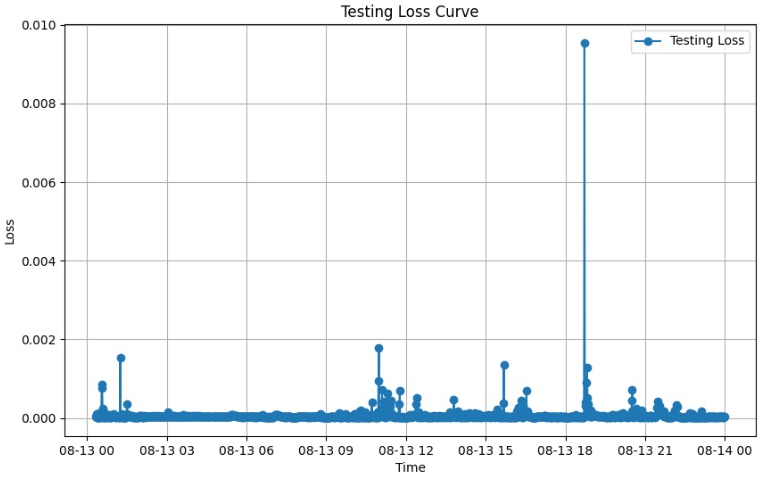

```
180 ost jobs and 125 mdt jobs are being tagged.
2025-08-21 20:49:00.589753 finished tagging in 0.80 s -- 0 anomalous jobs detected.
ost anomaly: 0, mdt anomaly: 0
```




图(1)异常数据损失




图(2)正常数据损失




图(3)跨文件系统测试

# 未来工作

- 技术优化
  - 通过改进模型结构，提升模型对隐蔽性异常的识别能力
- 引入新兴技术
  - 考虑数据在不同OSS上的分布，捕捉细粒度异常识别能力。
  - 结合边缘计算技术，在存储系统的边缘节点部署轻量化的监测模块
- 结合计算作业提交数据综合分析
  - 当前方法缺乏对用户的行为分析，针对大批量的小流量读写请求造成的异常检测能力有限，考虑与作业提交端数据结合获取更精细的异常检测能力。

# 感谢聆听！

侯思琦

中国科学院高能物理研究所计算中心

2025年8月25日 Monday