AHA：面向运营时间序列应用的可扩展替代历史分析

Harshavardhan Kamarthi 佐治亚理工学院 美国亚特兰大
hkamarthi3@gatech.edu

Harshil Shah Conviva 美国福斯特城
hshah@conviva.com

Henry Milner Conviva 美国福斯特城
hmilner@conviva.com

Sayan Sinha 佐治亚理工学院 美国亚特兰大
sayan.sinha@cc.gatech.edu

Yan Li Conviva 美国福斯特城
yan@conviva.com

B. Aditya Prakash 佐治亚理工学院 美国亚特兰大
badityap@cc.gatech.edu

Vyas Sekar 卡内基梅隆大学 美国匹兹堡
vsekar@ece.cmu.edu

arXiv:2601.04432v1 [cs.DB] 7 Jan 2026

摘要

许多运营系统会收集有关用户/系统在关键性能指标方面的高维时间序列数据。例如，互联网服务提供商、内容分发网络和视频传输服务会收集与元数据（如位置、设备、互联网服务提供商）相关的用户会话体验质量指标。对于这类历史数据，运营人员和数据分析师通常需要执行回溯分析；例如，分析异常检测算法、试验不同的警报配置、评估新算法等。我们将这类针对运维数据集的工作负载称为替代历史分析。我们表明，在此类场景中，传统的数据处理解决方案（如数据仓库、采样、草图化和大数据系统）要么运营成本高昂，要么无法保证准确回放。我们设计并实现了一个名为 AHA（替代历史分析）的系统，克服了这两项挑战，为高维数据提供成本效益和保真度。AHA 的设计基于对这类工作负载的分析性和实证性洞见：1）底层统计量的可分解性；2）属性值组合中活跃子群体数量的稀疏性；3）现代分析型数据库中聚合操作的效率结构。我们使用多个真实世界数据集，并结合一家大型视频分析公司生产流水线的案例研究，表明 AHA 对广泛的下游任务可提供 100% 的准确率，与传统方法相比，总拥有成本最高可降低 85×（即计算 +存储）。

本作品采用知识共享署名 4.0 国际许可协议授权。*KDD 2026*年，济州岛，大韩民国。© 2026年版权所有，归所有者/作者所有。ACM ISBN 979-8-4007-2258-5/2026/08 https://doi.org/10.1145/3770854.3783945

CCS 概念

•**信息系统** → **数据库管理系统引擎**；•**计算方法** → 监督学习；机器学习算法。

关键词

大规模数据处理、时间序列、摘要

ACM 参考文献格式：

Harshavardhan Kamarthi、Harshil Shah、Henry Milner、Sayan Sinha、Yan Li、B. Aditya Prakash 和 Vyas Sekar。2026年。AHA：面向运营时间序列应用的可扩展替代历史分析。载于第*32*届 *ACM SIGKDD* 知识发现与数据挖掘会议论文集 *V.1*（*KDD 2026*），*2026*年*8*月*9*日至*13*日，大韩民国济州岛。ACM，美国纽约州纽约市，12页。https://doi.org/10.1145/3770854.3783945

1 引言

许多运营系统会随时间收集有关用户和系统级性能指标的高维数据。分析人员需要在这些收集到的数据上运行各种算法；例如，针对按属性分组的用户子组进行异常检测。例如，大型视频监控系统会收集视频会话的“体验质量”指标（如比特率、缓冲），以发现影响用户子组的异常模式 [25, 30]。这些模式可能包括影响某个子组的性能下降；例如，来自某个互联网服务提供商与城市组合的用户是否出现了性能下降?

运营分析任务通常需要能够对纵向数据集执行替代历史分析 [1]。例如，机器学习科学家可能需要对历史数据集进行回归测试（即持续集成/持续交付），以进行基准测试 [27]。在异常检测中，客户可能会查询：如果使用不同的灵敏度阈值，几周前触发的警报是否会被抑制 [33]。

在运营环境中支持替代历史分析颇具挑战。数据集和分析涉及庞大的规模、高昂的成本、下游任务的不可预测性，以及数量呈组合爆炸式增长的感兴趣子群。事实上，几种看似自然的解决方案都难以奏效。因此，实现低总拥有成本和准确

解决方案	低成本的高维数据	高保真度性/灵活性	支持广泛的统计数据。
存储原始会话数据	No	Yes	Yes
跨会话采样	Yes	No	No
子总体统计量使用草图（例如 [30]）	Yes	No	No
键值存储 for 所有亚组统计（例如，[7]）	No	Yes	Yes
替代历史分-析（AHA）	Yes	Yes	Yes

表 1：传统解决方案无法满足支持运营时间序列任务的替代历史分析所需的关键要求。

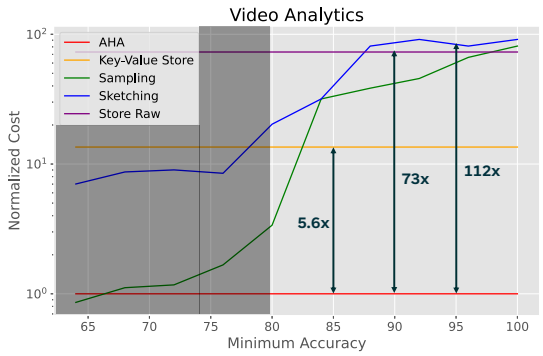


图 1：对于至少 90% 分位数的队列，达到给定最低准确率时各基线相对于 AHA 的成本。AHA 比最佳的 100% 准确基线高效 5.6 倍，比存储原始数据这一默认基线便宜 73 倍，并且为了达到 > 95% 准确率这一共同目标，其成本可能比近似解决方案低 100 倍以上。

针对未预见任务的回放一直难以实现。要了解原因，请看表 1 中的初步方案。一个极端做法是，将每个时间步的原始会话测量数据（即属性、指标）存储在数据库中，并在发出查询时计算感兴趣的组合和统计量。但遗憾的是，这会面临数据保留和成本方面的挑战；例如，每天可能产生数 TB 的原始用户会话数据。另一个极端做法是，预先计算少数属性组合（例如“重频项”）和感兴趣的统计量。类似地，另一种自然的做法是只存储触发的警报相关数据。然而，这些方法无法覆盖用户未来想要尝试的查询。

本文介绍了*AHA*的设计与实现。这是一个用于支持运营系统中替代历史分析的实用系统。AHA 的设计基于关于运营数据结构、查询模式、此类场景中通常使用的下游任务，以及现代数据处理系统能力的关键分析和实证洞见。

我们构建了一个实用系统，在数据摄取时跟踪多个（或所有可能的）子群体子集的指标所需统计量，并据此推导其他群组的指标。这样，当未来的替代分析历史任务下达时，我们便能计算所需的目标指标。这种解耦工作流使我们能够延迟绑定摄取时计算与查询时计算，从而实现较低的总拥有成本，并准确回放未预见的任务。总之，我们的工作做出了以下贡献：**(1)提出替代历史分析问题（§2）：**据我们所知，我们首次在运营时间序列场景中提出替代历史分析问题。我们确定了大规模支持这一能力所面临的关键成本与准确率挑战；**(2)设计并实现AHA （§4）：**我们设计并实现了 AHA，使其能够高效检索任何可能群组的大范围特征；**(3) 正确性与覆盖保证（§4）：**我们提供理论保证，证明在广泛的下游任务和方法中，AHA 都能达到完美准确率；**(4) 在真实部署场景中进行基准测试（§5）：**我们在多个数据集上对 AHA 及其他最先进的解决方案进行基准测试，其中包括在一家大型视频分析公司将其用于生产环境。结果显示，与基线相比，AHA 在准确率不受影响的情况下，总拥有成本降低了 34 至 85 倍，每月可节省超过 70 万美元。我们还开展了一项部署研究，观察其对生产数据管道的影响，发现总成本降低了 6.2 倍。

2 背景与动机

本节介绍运营环境中时间序列分析任务的结构，以及对替代历史分析的需求。我们将明确支持这些用例所需的条件，并探讨现有解决方案为何无法满足需求。

2.1 运营时间序列分析

我们研究运营环境中出现的时间序列分析工作流。从总体上看，运营环境涉及高维数据，运营人员关注跨多个用户／端点子群的时空洞察，以指导运营。为具体说明这一点，我们以部署了本系统的大型视频分析服务为例。视频分析服务会对用户数据进行体验质量（QoE）分析，帮助内容提供商和互联网服务提供商提升用户体验、观众留存率和满意度 [25]。系统会收集各类视频分析指标，这些指标来自由许多特征各异的用户构成的视频会话，包括比特率、丢帧数量等。每位用户的观看会话还会标注称为属性的附加元数据。

典型的分析工作流涉及识别指标中重要的模式，这些指标针对的是根据某些属性（如地理位置、互联网服务提供商、所用设备等）的相似性划分的用户群组 。例如，当来自某个州、使用特定互联网服务提供商的用户缓冲时间异常偏长时，可以通知该互联网服务提供商，以便其解决网络问题。检测 QoE 问题和模式，需要检测为各用户组生成的指标中的异常；这些用户组根据组内用户共同拥有的属性赋值集合来确定。典型数据集根据用户属性的数量及其可能的组合，包含数百万个可能的用户组。这些群组会针对多个指标进行监控，以发现其中的模式。

在网络分析、监控日志、使用情况

分析等其他领域中，也可以发现类似的用户和产品分析应用；这些领域会收集并分析多种类型用户的数据，例如电信、可观测性、物联网、移动端／广告分析等 [9, 13, 18, 19, 23, 31]。

2.1.1 数据集架构和典型查询模板。 我们以视频分析作为主要应用，定义用于子群体分析的数据架构（见图 2）。数据集由不同用户的会话 遥测数据组成，例如观看视频的会话。其他示例包括移动应用中的用户会话、网络流或 TCP 会话。每个会话都附有一组属性 ，用于描述用户或会话，例如用户位置、互联网服务提供商、设备信息等。每个会话在每个测量时间段都有一组每会话关键绩效指标 ，例如按分钟统计的 QoE 指标，如缓冲时间、比特率等。

会话根据其属性分组为队列 。队列是具有相同属性集合的一组会话。例如，队列可以包含来自特定位置的会话，或来自特定互联网服务提供商的所有会话。在我们的实验中，根据数据集的不同，观察到的队列数量从 300 个到超过一百万个不等。

运营任务通常希望以队列级粒度分析指标，以了解整体表现良好或不佳的模式。因此，我们可以聚合队列中所有会话的会话测量值（称为会话指标），以获得队列汇总指标（称为指标）。队列指标的示例可以是队列中所有会话的平均缓冲时间。

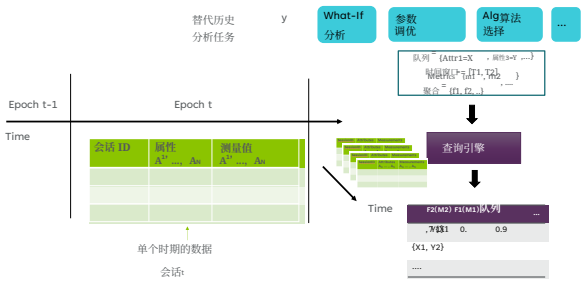


图 2：替代历史分析的数据设置。用户需要获取过去任意时间步上任意用户队列的聚合统计量。

2.1.2 对替代历史分析的需求。虽然此类运营数据用于实时检测异常和进行预测，但我们与运营人员、算法开发人员和机器学习运维从业者的讨论揭示了一个更为根本的痛点。本质上，由于工作负载变化、算法开发和客户需求，运营系统始终处于不断变化的状态。因此，许多用例要求运营人员和分析师在日常运营中访问并查询纵向数据。大多数查询都可以表述为一个函数：该函数根据某个队列的会话指标推导聚合统计量，并据此进行预测。

形式化地，令 $\mathbf{m}$ 为特定时间步 t 的会话指标向量。 $F(\{\mathbf{m}_i\}_{i\in S})$ 是函数 F，用于计算队列 S 中各会话指标的聚合统计。我们将具有属性 $\mathrm{A} = \mathrm{a}$ 的队列定义为队列 S。例如，使用 Spark 时，我们可以编写 PySpark 查询来计算此统计量；若要为所有队列计算此度量，可以对所有可能的队列运行此查询：

```
去重_的 = 会话.select("attributes").distinct()
对于 一个_值在 队列_列表:
    已筛选_数据框 =会话.筛选(会话.属性 ==
        一个_值["attributes"])
    result_df = filtered_df.select(
        F.col("属性"),
        F.expr("F(mi_列)")              # 应用函数 F,
            将 mi_column 替换为实际列
    )
```

有时，我们会按时间聚合指标：

```
去重_一个 = 会话.select("attributes").distinct()
对于 一个_值在 队列_列表:
    已筛选_数据框 =会话。过滤(
        (会话.时间 >t1) &
        (会话.时间 <t2) &
        (会话.属性 ==一个_值["attributes"])
    )
    result_df = filtered_df.select(
        F.col("属性"),
        F.expr("F(mi_列)")              # 替换        'F(mi_column)'
            为你实际应用的函数
    )
```

我们将示例应用描述如下：

- 假设分析： 运营时间序列分析需要为用户和分析人员提供配置选项，例如，决定异常应持续多久，或受到影响的用户数量。然而，客户需要数据驱动的指导，以了解为何会生成预测，或更改算法参数将如何影响基于历史数据的预测准确率、假阳性率等。例如，函数 F 可以测试某个测量值的统计量（如应用的平均登录时间）是否大于一个固定阈值。更改阈值或统计量，以检查某些预测是否发生变化，是一种常见用例。

- 面向机器学习运维的持续集成/持续交付中的以数据为中心的回归测试：鉴于工作负载和模型漂移不断变化，机器学习工程师和算法开发人员需要改进算法和配置；例如，随着工作负载或基线变化，进行超参数调优。为此，他们需要采用相当于开发运维最佳实践的做法，例如回归测试。例如，回归测试可以是在 F 中测试算法，以便根据历史数据为新数据集优化准确率，找到更合适的阈值和参数。

- 算法选择：随着新的时间序列方法不断涌现，机器学习团队希望持续测试研究社区提出的新颖方法。然而在实践中，没有一种方法适用于所有情况，机器学习团队需要在部署前，针对各自的具体场景（例如：启动视频会话的故障率），严格测试新的算法方法以及特征工程。通过更改函数 F 并调整模型参数来测试新算法和统计量，可以帮助建模人员发现性能更佳或效率更高的工作流。

3 问题表述

基于这些示例场景，我们正式定义运营时间序列应用中的替代历史分析问题。在推导关键需求之前，我们先介绍一些符号表示和定义。随后，我们将讨论为什么文献中的典型解决方案无法满足这类运营问题的需求。

用户组和属性：我们正式定义数据集的重要方面。将时间 t 处的数据集记为 D_t。该数据集由每个时间步（分钟或秒）多个用户（通常为数百万或数千万）的数据组成

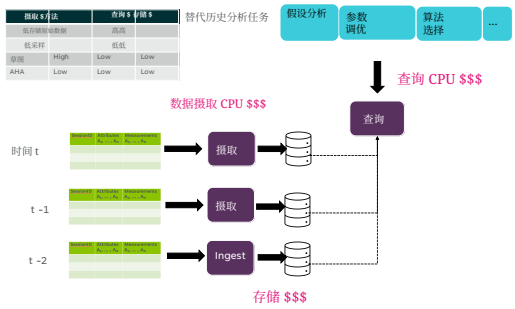


图 3：替代历史分析的设计空间：会话数据在每个时间时期被摄取并存储。用户根据不同任务查询跨时间存储的数据。AHA系统根据特定应用，从存储的摘要中计算所需特征。

。每个用户都标注有 M 属性，用于描述与用户相关的特征。在视频分析场景中，这些属性可以是用户位置、互联网服务提供商、设备信息等。形式上，数据集中的每个用户 j 由 M 个属性描述，这些属性初始化为 $Attr_j = \{Attr_j[i]\}_{i=1}^{M} \in \mathrm{A}_1 \times \mathrm{A}_2 \times \cdots \times \mathrm{A}_M$，其中 A_i 是属性 i 所有可能取值的集合。我们作出一个合理的假设：每个属性只能取离散值。此外，我们还记录每个用户的 K 指标。用户 j 的指标记为 $\mathrm{m}j = \{m_j[i]\}_{i=1}^{K}$。因此，时间 t 处的数据集为 $\mathrm{D}_t = \{(Attr_j, \mathrm{m}j)\}_{j=1}^{N_t}$，其中 N_t 是时间 t 跟踪的用户数。每个用户组由该组所有用户共有的属性定义。群组 $C(\mathrm{a})$ 由属性初始化 $\mathrm{a} \in \mathrm{A}_1 \cup \{*\} \times \mathrm{A}_2 \cup \{*\} \times \cdots \times \mathrm{A}_M \cup \{*\}$ 定义，即 a 是属性值序列，或 $*$，其中 $*$ 表示群组用户可以取任意值。我们将 C_t 记为 D_t 中的非空群组集合。然而，可能的用户组数量随属性数呈指数增长：用户组数量等于属性可能子集的数量与每个属性可能取值数量的乘积（$\prod_{i=1}^{M}(|\mathrm{A}_i| + 1) - 1$）。因此，我们利用对数据集和分析需求的重要见解来设计 AHA，使其具备成本效益。

分析算法： 对于每个用户组 $C(\mathrm{a})$，令 $\mathrm{D}_{t,\mathrm{a}} = \{(Attr_j, \mathrm{m}j) : Attr_j \in \mathrm{a}\}$ 表示该群组中所有用户数据的集合。具有配置参数 θ 的算法 $Alg(\mathrm{D}_{t,\mathrm{a}}; \theta)$ 会生成输出 $Alg(\mathrm{D}_{t,\mathrm{a}}; \theta) \in \mathbb{R}^b$。例如，对于检测给定队列是否异常这一任务，$Alg(\mathrm{D}_{t,\mathrm{a}}; \theta) \in [0, 1]$，即预测给定群组为异常的概率。请注意，$\mathrm{D}_{t,\mathrm{a}}$ 包含来自数量不定的用户的指标数据；也就是说，任一群组在不同时间观察到的用户数量会有所变化，而且许多群组在大多数时间步中观察到的样本都很少或为零。我们可以将算法 $Alg()$ 分成两部分。首先，从数据集 $\mathrm{D}_{t,\mathrm{a}}$ 中提取所需的固定特征集 $F(\mathrm{D}_{t,\mathrm{a}}; \theta_F)$。这些特征是根据群组中所有用户的指标汇总得到的各种统计量。通常，领域专家会结合应用需求或领域知识设计这些特征。随后，使用这些特征输入分析模型 $\mathrm{M}(F(\mathrm{D}_{t,\mathrm{a}}); \theta_M)$，为群组 a 生成预测。特征模型 M 可以是任意统计模型或机器学习模型。因此，该算法可描述为 $Alg() = < F, \mathrm{M}\ \theta >$，其中 $\theta = (\theta_F, \theta_M)$。

需求： AHA 的目标是支持类型为 $< \mathrm{C}\ Alg()\ \theta,\ T >$ 的查询 Q，该查询针对特定群组 G，利用带有超参数 θ 的潜在新算法 $Alg()$ 生成输出。概念上，我们将候选方案分为三个阶段：（1）摄取并汇总多群组数据；（2）存储汇总后的数据，以支持广泛的回溯分析；（3）从数据摘要中提取分析算法 $Alg()$ 所需的相关特征/信息。

我们为任何 AH 方案定义以下设计目标：

- **未来分析的估计等价性：** 给定一组异常检测算法 $Alg() \in \mathrm{Alg}$，(IngestReplay, FetchReplay) 提供任务级等价性，当

$$Alg(F(\mathcal{D}); \theta) \approx Alg(F(Repl(\mathcal{D})); \theta) \forall \theta, \forall Alg() \in \mathrm{M} \qquad (1)$$

也就是说，如果我们使用会话数据，模型的估计结果应与使用回放中的特征所得的估计结果相似。

- **队列指标的等价性：** 类似地，当 $F(\mathrm{D}) \approx F(Repl(\mathrm{D}))$ 时，(IngestReplay, FetchReplay) 提供队列级等价性。注意：能够保证 100% 准确率的方法称为强等价 解决方案。其中包括键值存储等确定性解决方案，以及能够准确计算指标的方法等。相比之下，采样和草图化只能提供近似估计。这类方法称为弱等价 解决方案。

- **低总拥有成本**：总拥有成本取决于摄取和回放的计算成本以及存储占用。理想情况下，回放数据的大小（记为 $Repl(\mathrm{D})$）应比原始数据集 D 小几个数量级：$|Repl| << (|\ |.$ D) D。使用 IngestReplay 和 FetchReplay 为模型推导特征的计算成本，应与直接从原始数据 $F(\mathrm{D})$ 推导特征的成本相近，或显著更低。

相关工作的局限性： 为应对这一挑战，我们考察了文献中一些看似自然的稻草人方案，并讨论它们为何无法满足我们的关键要求。总体而言，我们可以根据摄取时和摄取 以及查询时所执行的计算，将选项的设计空间概括为图 3 所示的形式。在这一框架下，我们可以将以往针对多维数据摄取与检索的工作分为近似方法和精确方法。根据在摄取/查询时执行的精确或近似分析类型，我们需要在摄取、存储和查询时的成本之间进行权衡。

精确方法：大多数标准运维流水线要么在摄取时从数据流中计算所需特征，要么在之后获取数据时再计算。这些方法大多使用 Spark、Hadoop 等数据库分析解决方案 [6, 16, 38]。这些 SQL 和 NoSQL 数据库通常会精确计算所需特征，因此可提供 100% 的准确率。一种极端做法是在摄取时不做任何处理，只存储原始会话数据，并在查询时执行精确查询。另一种做法是针对每个可能的子总体，对多个子组的聚合统计进行汇总，这或许能加快计算速度，但为了支持任意查询，摄取时仍需存储全部原始数据。随着待存储数据量增加，或随着属性数量增加、用户组数量呈指数级增长，这两种方案的成本都会很高，正如我们的实际应用所示 [4]；尽管其中一些数据库采用了多种无损压缩方法。

近似方法：这些方法通常在摄取或查询时计算某个度量的近似值，从而降低计算成本和

内存成本。在线采样是一种常用技术，可降低计算和延迟成本[11, 35]。基于草图的方法以次线性复杂度在内存与准确率之间提供更高效的有界权衡。这些框架在摄取过程中生成近似摘要，并利用这些摘要估算统计数据，同时提供先验可证明的误差界[14, 17, 30, 38]。Apache Druid 等分析型数据库[38]也使用这些近似算法来计算不同元素、直方图等。虽然这些方法可以显著节省存储成本，但无法保证样本不足的用户组及相关场景具有预测等价性，并且在摄取大型高维数据集时可能需要大量计算资源。

4 AHA设计与实现

本节将介绍关于数据集特性及大多数应用中分析算法的关键分析性洞见，并通过实证验证这些洞见。我们利用这些洞见，为构建 AHA 提供实用依据，使其能够在大规模场景下实现高效性和准确率。

4.1 主要洞见

替代历史分析查询中，候选统计量和感兴趣的子群都难以预测。因此，从系统角度来看，我们的设计理念是采用延迟绑定架构。我们认为，通过延迟绑定子组摘要和所支持的任务，AHA 将更具成本效益且更易扩展。基于这一总体设计决策，我们聚焦于分析任务和数据集的特性，以识别有助于 AHA 有效存储并高效检索用户子组相关数据的实用属性。

洞察 1 任务可分解性：对于许多下游分析和异常检测任务，所需的统计量 $F(\cdot)$具有**可分解的**性质，即“父”用户组的统计量可以通过构成该父组的“子”群组的少量统计量推导得出。

在运营时间序列场景中，相关用户组之间存在子集关系。例如，包含使用特定互联网服务提供商（如 Comcast）的用户的用户组，在一次视频会话中可能使用多个内容分发网络。此外，大多数分析算法所需的必要统计量F可以通过统子组的统计量。例如，用户组的某项度量的均值可以由各个子组的该度量均值推导得出。这是一种众所周知的性质，称为可分解性（见 §4.3）。因此，在许多情况下，我们可以根据从各州用户组收集的统计量，为美国用户的“父”组生成分析算法 F 所需的统计量。这一洞见使我们能够：1）避免使用需要存储所有可能子组数据的高存储开销解决方案；2）通过针对用户组的回放算法 IngestReplay 缩小所存储统计量的范围；3）选择需要存储统计量的用户组。

形式上，给定会话数据集 D_t，预测模型的最终输入是一组针对所有可能子组C_t的特征。C_t的大小上限为 $O(\prod_{i=1}^{M}|\mathrm{A}_i|)$，这是属性 M数量的指数级函数。不过，我们可以为覆盖所有可能子组的最小子组集合生成必要统计量 F'，该集合称为 **LEAF**。这些子组中，所有

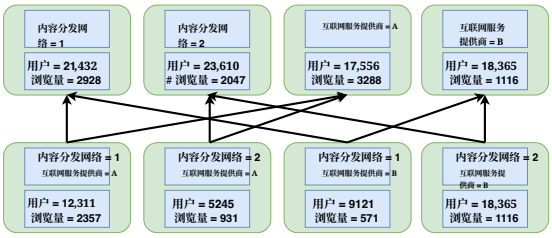


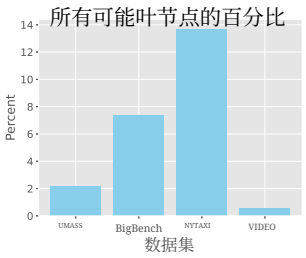
图 4：可分解性：群组的特征可以由子组的特征推导得出。

属性都初始化为特定值（即 $Leaf(\mathrm{D}) = \{C(\mathrm{a}) : \mathrm{a} \in \mathrm{A}_1 \times \mathrm{A}_2 \times \cdots \times \mathrm{A}_M\}$）。令 $\mathrm{A}_t \subseteq Attr_1 \times Attr_2, \ldots\ldots, \times Attr_M$因此，AHA系统分两个阶段运行：

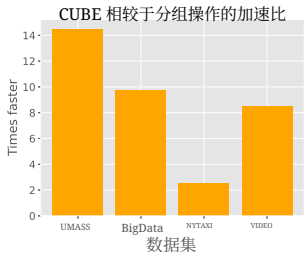
- IngestReplay 将叶组的必要统计量 F，存储在回放存储中。

$$Repl(\mathcal{D}_t) = \texttt{IngestReplay}(\mathcal{D}_t) = \bigcup_{\mathrm{a} \in \mathcal{A}_t} F'(\mathcal{D}_{t,\mathrm{a}}). \quad (2)$$

- FetchReplay 根据中间特征 $\bigcup_{\mathrm{a}' \in Child(\mathrm{a})} F'(\mathrm{D}_{t,\mathrm{a}'})$，为任意用户组$C$（a）推导所需特征 F，其中 $Child(\mathrm{a})$ 是包含在子组 a 中的叶组集合。



(a) 观测到的叶组数量少于可能的最大叶组数量



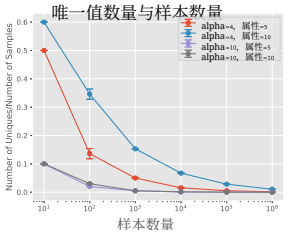
(b) CUBE 操作在所有数据集上的速度比 GroupBy 分组操作快 3 到 14 倍

图5：子组稀疏性（洞见 4.1）和 CUBE 效率（洞见 4.1）的证据

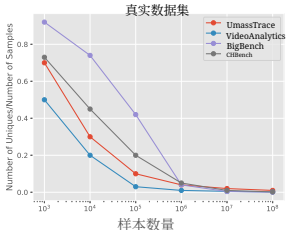
洞见2 活跃子组稀疏性带来的单节点驻留：尽管可能的子组数量会随着属性数量 M呈指数级增长，但在任何会话中，给定时间步观测到的**活跃子组**数量都要少得多。

然而，即使有上述洞见 4.1，叶组的可能总数仍可能随属性数量呈指数级增长。实践中，我们发现，任一给定会话中出现的叶组数量要少得多，因此在许多情况下，可以将其存储在单个节点的内存中。事实上，任一给定会话中出现的活跃用户会话数量通常也可能适合存储在单个节点的内存中。这使我们能够通过存储数量少得多的 LEAF 子组所需的信息，将工作流解耦。此过程通常需要一个单节点系统，以便在 IngestReplay 和 FetchReplay 期间高效处理数据，而无需

多处理器通信带来的开销。因此，我们仅存储会话数据集中观测到的 LEAF 群组：$A_t = \{a \in Attr_1 \times Attr_2, \ldots, \times Attr_M || C(a)| > 0\}$。观测到的叶组相对于可能的最大叶组数量的平均比例（图 5a）在各数据集中为 0.5% 至 13.7%，足够小，可以存储在具有 64GB 内存的机器中。



(a) 合成数据中，遵循齐夫分布时，不同 α 值和属性数量对应的唯一属性值相对比例



(b) 不同样本大小下，真实世界数据集中随时间观察到的唯一群组比例

图 6：随着样本大小增加，唯一会话（LEAF）的数量增长得慢得多。我们绘制了合成数据和真实数据中唯一值占总样本大小的相对比例，并观察到相似的比例递减模式。

洞见 3 *CUBE* 操作对内存中的数据十分高效：当数据集能够放入单台机器的驻留内存时，在传统分析型数据库中使用 CUBE 操作，其性能可能优于在 ***Spark*** 等通用计算引擎中针对每个子组运行 GROUPBY 操作的工作流。

按需计算子组统计量的典型解决方案，是在多节点集群上通过 GroupBy 类型的操作（例如使用 Spark 或 SQL）来实现。不过需要注意的是，在 AHA 中，我们之前的洞见 4.1 表明，可以将所有叶组的必要统计量存储在单个节点上。在这种情况下，我们可以使用更高效的 CUBE 操作符 [20]。SQL 及其他数据处理系统中的“CUBE”操作符是一种特殊的 GROUP BY 功能，可为选定列中的值生成所有组合的聚合值。大数据库和大数据文献中对此已有深入研究 [26, 36]。事实上，现代 OLAP 引擎（例如 ClickHouse）原生支持 CUBE 操作。虽然在数据移动量较大的分布式环境中，CUBE 操作可能成本高昂，但在单节点内存驻留环境中，可以根据叶组的中间统计量，高效地推导出所有父组的属性值的所有可能组合所需的统计量（图 5b）。

4.2 设计与实现

结合这些见解，我们为 AHA 设计了以下工作流，包含两个逻辑阶段（图 7）：(1) 在摄取时，我们计算每个*LEAF*中与任务相关的不同感兴趣度量统计量。(2) 在查询时需要进行替代历史分析时，我们会对每个 LEAF 的统计量运行按时期划分的*CUBE*。假设我们需要为预测算法计算统计量 F

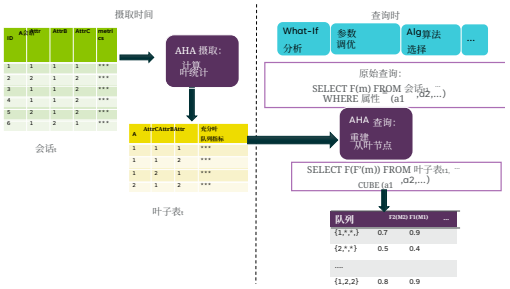


图 7：AHA 在摄取期间仅为少量叶子队列计算必要的指标，之后通过高效的 CUBE 操作为其他队列计算所需的指标。

适用于所有队列。如中所述，等效的 SQL 查询为：

$$\text{SELECT } F(\{\mathbf{m}_i\}_{i\in S}) \text{ FROM 会话 WHERE 属性 } = \mathbf{a} \tag{3}$$

对于所有可能的队列 $\mathbf{a} \in \in A_1 \cup \{*\} \times A_2 \cup \{*\} \times \cdots \times A_M \cup \{*\}$，其数量随属性 M 呈指数级增长。接下来，我们将按照 AHA 提出的方法，把这一系列查询转换为一个更高效的两步流程，从而显著降低存储和计算成本。

在摄取期间计算 *LEAF* 指标。我们根据会话指标数据计算 F' 所有叶子子组所需的统计量。相应的查询为：

$$\begin{aligned}&\text{CREATE VIEW } \texttt{leaf-table} \text{ as SELECT } F'(\{\mathbf{m_i}\}_{i\in S})\\ &\text{FROM sessions GROUP BY } (\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_M)\end{aligned} \tag{4}$$

该查询汇总了$_2$、.....、$\{A_1, A_M\}$ 在会话中观察到的所有队列。由于其数量远小于可能的队列数量（洞见 4.1），因此生成的汇总数据集也小得多。

在预测期间，根据 *LEAF* 指标为任意队列生成指标。我们使用叶子表中叶组的指标来计算 F' 任意队列的充分统计量。对于所有可能的队列，我们可以使用大多数数据处理平台（如 ClickHouse 和 Spark ）提供的 CUBE 操作，如下所示：

$$\text{SELECT } F'(F(m)) \text{ FROM } \texttt{leaf-table} \text{ CUBE } (\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_M) \tag{5}$$

或者，我们也可以使用许多数据库中实现的分组集，选择属性 $A'_j \subset \{A_1, A_2, \ldots, A_M\}$ 的 $F'$for select 组合，如下所示：

$$\begin{aligned}&\text{SELECT } F'(F(m)) \text{ FROM } \texttt{leaf-table} \text{ GROUP BY}\\ &\text{GROUPING SETS } (A'_1, A'_2, \ldots)\end{aligned} \tag{6}$$

由于 OLAP 平台对 CUBE 和分组集操作进行了高效实现，这一步比遍历队列的所有可能组合更快（洞见 4.1）。[1]。

4.2.1 实时部署详情。上述洞见促成了 AHA 在数据工程流水线中的成功部署：来自主要流媒体服务提供商的视频会话产生的大量原始数据被实时摄取。该系统用于核心数据处理流水线和告警系统，以识别所有可能用户子组的时间序列中的异常，并确定受异常影响最大的

[1]匿名代码：https://anonymous.4open.science/r/AHA_KDD25-3B63/

子群体及关键属性。该系统使用 ClickHouse 作为数据处理引擎和数据库，凭借其在 CUBE 等关键操作上的高效性满足需求。

4.3 AHA 的可靠性

我们将刻画利用本设计能够推导出的统计量类型。对于任意用户在任意给定时间的指标数据，它都是集合 M 的一个元素。用户子组的数据是 M 的一个有限子集。统计量是一个函数 f：$2^{\mathrm{M}} \rightarrow \mathrm{O}$，它将子组的指标数据映射到输出域 O。例如，我们将均值统计量定义为 $mean = \frac{1}{|C|}\sum_{m\in C} m$，它映射为一个标量。

定义 1（自可分解统计量）。设 f 为一个统计量。若对于任意有限集合 $\mathbf{M}_0 \in 2^{\mathrm{M}}$ 以及 $\mathbf{M}_0$ 的任意有限不相交划分 $\{\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_N\}$（即，对于任意 $1 \le i < j \le N$ 和 $\bigcup_{i=1}^{N}\mathbf{M}_i = \mathbf{M}_0$，有 $\mathbf{M}_i \cap \mathbf{M}_j = \phi$），均满足以下条件，则称 f 为自可分解的：

$$f(\mathbf{M}_0) = f(\{f(\mathbf{M}_1), f(\mathbf{M}_1), \ldots, f(\mathbf{M}_N)\}). \quad (7)$$

例如，求和是一种自可分解统计量，用于对给定子组的指标求和。显然，子组中某项指标的总和可以通过将该子组各个互不相交分区中该指标的总和相加得出。

定义 2（可分解统计量）。 f相对于一组统计量 $N(f) = f_1, f_2 \ldots, f_k$ 是可分解统计量，当且仅当存在一个函数 A_f，使得对于任意有限集合 $\mathbf{M}_0 \in 2^{\mathrm{M}}$ 以及将 $\mathbf{M}_0$ 划分为任意有限个互不相交子集 $\{\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_N\}$ 的情况，均有：

$$f(\mathbf{M}_0) = A_f(\{f_1(\mathbf{M}_i)\}_{i=1}^{N}, \{f_2(\mathbf{M}_i)\}_{i=1}^{N}, \ldots, \{f_k(\mathbf{M}_i)\}_{i=1}^{N}) \quad (8)$$

简单来说，可分解统计量是指可以由其各分区的某些统计量通过函数计算得出的统计量。例如，均值可以通过各分区中数值的总和以及计数（即各分区的基数）计算得出。可分解统计量涵盖了运维流水线中的众多统计量，也包括我们部署应用中的统计量。AHA 支持广泛的预测分析，可处理均值、总和、极差、方差和其他阶矩等可分解统计量，这些统计量常用于各类算法。重要的不可分解指标则可以通过精度良好的近似算法计算得出。例如，我们可以计算直方图草图；这些可分解的草图会存储在每个叶组中，可用于计算多种统计量。中位数和其他四分位数可以近似估算 [21]。它们能够在分布式系统中预先计算来自不同分区的中间结果 [24, 39, 40]，从而减少节点间的通信量，并加快这些函数的实现；在 OLAP 数据库等直接查询原始数据成本高昂的场景中，这一点尤为有用。此外，函数 A_f 通常由加法、乘法等简单聚合函数以及平方根、对数等初等函数组合而成，这些函数在现代数据库中都能得到高效实现。

具体而言，我们推导出覆盖所有可能子组的最小子组集合所需的必要统计量，该集合称为**LEAF**。这些子组中的所有属性都初始化为特定值（即 $Leaf(\mathrm{D}) = \{C(\mathbf{a}) : \mathbf{a} \in \mathrm{A}_1 \times \mathrm{A}_2 \times \cdots \times \mathrm{A}_M, |C(\mathbf{a})| > 0\}$）。回放存储由 $Repl(\mathrm{D}_t) = \{(\mathbf{a}\ F_\prime(\mathrm{D}_{t,\mathbf{a}})) : \mathbf{a} \in Leaf(\mathrm{D}_t)\}$ 组成。这里，$F_\prime$是算法特征 F 的必要统计量集合：$F_\prime = \bigcup_{f\in F} N(f)$。这使我们能够在大幅降低实际存储成本的同时，提供预测等价性完美的理论保证。

定理1。 如果异常检测算法$\mathcal{M}$ 需要特征 F，且这些特征都是可分解的，那么我们只需在回放存储中存储叶子子组的必要统计量 F'，即可实现完美的预测等价性。

5 评估

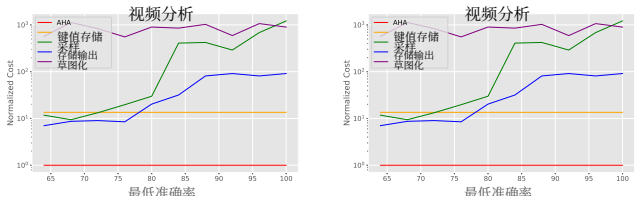
基线： 我们将 AHA 与以下替代数据处理解决方案进行比较。其中包含的强等价解决方案有：• StoreRaw：一种常见解决方案，在摄取过程中存储完整的原始数据，并在预测期间获取所需特征。

•KeyValueStore：与 StoreOutput 类似，我们仅通过哈希映射存储输出，该哈希映射将群组的属性映射到已存储的指标。我们通过扩展 Rust 编程语言中的哈希映射实现来实现键值系统。弱等价解决方案包括：• 采样：对于每个时间步，我们从用户样本中采样一小部分，并在获取时从这些样本中获取所需统计量。• 草图化：我们还使用一种先进的草图化解决方案*Hydra* [30]，该方案针对子群体分析而设计，使用通用草图来汇总统计量。

数据集： 我们在来自不同应用的多个数据集上评估 AHA。附录表 2 讨论了这些数据集的统计信息。• VideoAnalytics：在一家大型视频流分析服务提供商的实时部署中观测到的真实用户视频分析数据集。• UmassTrace：来自 [2] 的网络跟踪数据集。• NY- Taxi：包含纽约出租车和豪华轿车委员会提供的上下车时间及其他统计信息 [3]。• CH- Bench：由 ClickHouse 提供的匿名化数据库日志跟踪数据

• MGBench：来自 [8] 的合成基准测试，最初旨在测试图形处理器在数据 I/O 和简单计算工作流方面的性能 • IMDB：来自 IMDB 的数据集，包含电影列表及年份、演员、语言等多种特征。**基准测试详情**我们分析 AHA 和基线方法支持标准预测流程的能力。我们考虑生产环境中使用的异常检测算法：标准差阈值、 k 近邻（KNN） [5] 和孤立森林（IsoForest） [28]。所有这些算法都使用跨用户观测到的指标均值作为特征。

我们通过测量聚合指标的准确率（指标准确率）和异常检测算法的准确率（任务准确率），来衡量各个基线的保真度。指标准确率以均方根误差分数衡量，任务准确率则以异常分类准确率衡量。请注意，此处的准确率指的是，基于给定系统得出的预测与使用原始数据得出的预测之间的接近程度。精确计算指标的方法（包括 AHA）准确率为 100%，而近似方法的准确率可能较低。为确定系统的总成本（计算和存储），我们在 AWS 上使用配置相近的 EC2 系统，测量该系统在一个月内执行此操作的成本，并使用 S3 存储各个解决方案的汇总数据。有关设置、成本模型以及各基线参数设置的更多详情，请参见附录 §B。



（a）成本与指标准确率 (b) 成本与任务准确率

图 8：不同最小任务准确率和指标准确率要求下的归一化成本。对于指标准确率要求超过 80% 的所有数据集，AHA 在强等价解决方案中效率高出 55–130 倍，且是成本最低的解决方案。AHA 在所有任务准确率要求下成本均最低（> 90%），成本低 8.6–87 倍。

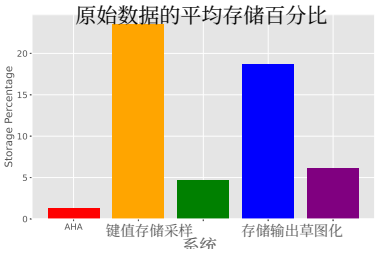


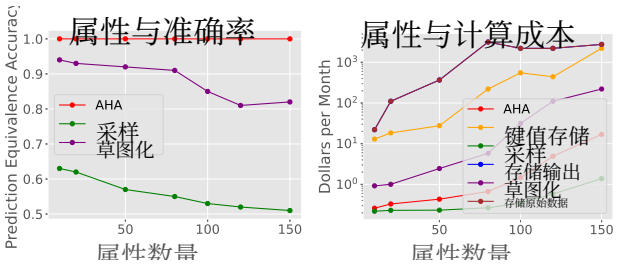
图 9：原始数据的存储占比。与其他解决方案相比，AHA 的存储需求最低。

5.1 结果

准确性与成本权衡：我们在图 1 和图 8 中展示了 VideoAnalytics 的成本与准确率（保真度）权衡，其他方案的结果见附录图 13。首先，正如预期，AHA、StoreRaw、KeyValueStore 和 StoreOutput 的保真度均为 100%。采样的平均准确率为 71%，而草图化的平均准确率为 88%。此外，近似方法在不同群组和时间上的准确率方差较大。采样和草图化的第 90 百分位平均准确率分别为 27% 和 53%。与存储原始数据或输出特征等标准解决方案相比，AHA 的成本低 34 至 85 倍；与草图化相比，成本低 6 至 10 倍。由于每小时的计算成本高于每 GB 每月的存储成本，存储成本比计算成本低 3 至 4 个数量级。我们在图 9 中测量了存储量的平均减少幅度。

AHA 的可扩展性AHA：我们从两个方面研究 AHA 和其他解决方案的可扩展性。首先，我们研究子组数量和用户属性数量增加所带来的影响。这会增加叶组的数量以及 CUBE 操作的复杂度。近似方法还包括增加群组数量，同时减少每个群组的样本数量。我们使用合成数据 Synth，其中的属性按照 Zipf 分布生成，以模拟 VideoAnalytics 中属性的平均分布。我们保持用户样本数量与 VideoAnalytics 相同，并任意增加属性数量

以模拟一种常见于视频分析的情况：我们可以访问更多用户属性。



(a) 准确率可扩展性 (b) 成本可扩展性

图 10：AHA 在增加属性时可扩展成本，同时保持准确率完美。

随着数量增加，近似方法的准确率会降低在属性方面（图10a），这是由于低采样群组数量增加所致。其计算成本比其他完全准确的解决方案低两个数量级，也显著低于基于草图的解决方案。AHA 可随着工作负载数量的增加进行水平扩展。我们通过多个并行工作负载来衡量其可扩展性，例如需要同时为许多客户执行分析时的情况。随着并行工作负载数量增加，并测量每个节点上运行各任务时的总成本。与其他完美回放解决方案以及草图解决方案相比，AHA 在工作负载数量增加时具有更好的扩展性（图11）。

5.2 部署经验

我们通过详细研究生产环境中的一个核心应用，展示 AHA 的实际有效性。该应用涉及对一款用于流式传输视频数据的移动应用进行用户旅程分析。本研究评估了预处理和分析性能，并跟踪用户导航模式。

设置： 虽然完整的生产数据集需要每分钟摄取并处理数百 GB 的数据，但我们关注的是其中一个子集，重点分析 30 天内测量的 10 项用户属性。我们分析了一个为期 30 天的生产数据子集，重点关注 10 项用户属性。该子集包含 26.17 GB 的日志，涵盖 95,408,549 个用户会话。我们评估三项关键的用户参与度指标：(i) 成功的跳转操作，定义为用户主动跳转后恢复播放；(ii) 在首页上播放的推荐视频数量；以及

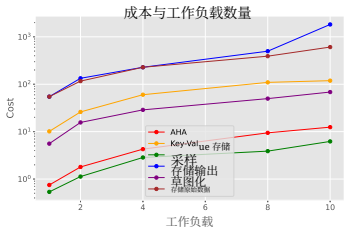


图11：随工作负载数量变化的成本。AHA 能够随工作负载数量的增加高效扩展。

（iii）点击的轮播推荐项数量。体验质量（QoE）度量统计这些成功操作的次数，并在预处理期间按分钟分组。由于我们关注的度量是每个用户组的操作总数，因此我们将各会话中的操作次数相加——这是一种 AHA 可优化的可分解操作。与基线在原始数据上使用 GROUP BY 反复执行单一用途的聚合不同，AHA 采用两阶段方法：先创建包含可分解统计量（求和 +计数）的 LEAF 表，再基于预先计算的统计量高效地执行类似 CUBE 的汇总，而无需重新扫描原始数据。

性能比较。 AHA 在每分钟数据聚合方面实现了 1.25× 倍加速，同时生成完全一致的输出（45,454 个队列）。对于全部数据，AHA 在 2:30 小时内完成了该操作，而基线需要 3:08 小时。对于使用 QoE 指标进行的下游回归分析，AHA 在预处理后的数据上展现出显著的性能提升。两种方法实现了完全一致的模型质量（$R^2 = 0.97$），但 AHA 优化后的数据表示使查询执行速度提升了 6.2× 倍。在扩展到如此大规模的数据集时，这一性能提升至关重要。本示例验证了 AHA 在保持分析准确性的同时，能够在分析流水线的数据预处理和查询执行阶段提供显著的性能提升。

6 结论

运营时间序列问题具有数据量大、基数高且持续变化等特征，迫使从业者在回顾性分析中牺牲成本效益或保真度。本文针对这些挑战提出了一种实用且严谨的解决方案：AHA。借助数据的结构特征、查询工作负载和现代分析型数据库，AHA 提供了一种切实可行的替代方案，使我们能够兼顾成本效益和完美准确率。我们的实验和案例研究表明，AHA 在准确率不受任何损失的情况下，将总拥有成本降低了 34 至 85 倍，相当于每月节省超过 70 万美元。我们观察到，已部署的重要应用的成本节省约为 75%。我们相信，我们的工作是识别并解决替代历史分析这一实际运营问题的第一步，也是重要的一步。

致谢：本研究部分由美国国家科学基金会（Expeditions CCF-1918770、CAREERIIS-2028586、Medium IIS-1955883、Medium IIS-2403240、Medium IIS-2106961、PIPP CCF-2200269）、美国国立卫生研究院（1R01HL184139）、美国疾病控制与预防中心 MInD 计划、Meta 教师捐赠、杜比捐赠，以及 Conviva 和佐治亚理工学院提供的资金与计算资源支持。

参考文献

[1] [无日期]。机器学习运维：机器学习中的持续交付与自动化流水线。https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning。[2] [无日期]。网络 - 马萨诸塞大学跟踪记录库。https://traces.cs.umass.edu/index. php/network/network。（访问日期：2023年11月20日）。[3] [无日期]。TLC 行程记录数据 - TLC。https://www.nyc.gov/site/tlc/about/tlc-trip- record-data.page。（访问日期：2023年11月20日）。[4] Sameer Agarwal、Barzan Mozafari、Aurojit Panda、Henry Milner、Samuel Madden 和 Ion Stoica。2013年。BlinkDB：针对超大型数据提供误差有界、响应时间有界的查询。载于第 *8* 届 *ACM* 欧洲计算机系统会议论文集。29–42。[5] Mennatallah Amer 和 Markus Goldstein。2012年。基于最近邻和聚类的 RapidMiner 异常检测算法。载于第 *3* 届 *RapidMiner* 社区会议与大会（*RCO MM 2012*）论文集。1–12。

[6] Michael Armbrust、Reynold S Xin、Cheng Lian、Yin Huai、Davies Liu、Joseph K Bradley、Xiangrui Meng、Tomer Kaftan、Michael J Franklin、Ali Ghodsi 等。2015。Spark SQL：Spark 中的关系数据处理。载于《2015 年 ACM SIGMOD 国际数据管理会议论文集》，1383–1394。[7] Ran Ben-Basat、Gil Einziger、Roy Friedman 和 Yaron Kassner。2016。数据流和滑动窗口中的重频项。载于《IEEE INFOCOM 2016——第 35 届 IEEE 国际通信会议》，IEEE，1–9。[8] T Ben-Nuun。2017。MGBench：多 GPU 计算基准套件（CUDA）。[9] Paul Bergmann、Michael Fauser、David Sattlegger 和 Carsten Steger。2019。MVTec AD——用于无监督异常检测的综合真实世界数据集。载于《IEEE/CVF 计算机视觉与模式识别会议论文集》，9592–9600。[10] Ane Blázquez-García、Angel Conde、Usue Mori 和 Jose A Lozano。2021。时间序列数据中的离群值/异常检测综述。《ACM Computing Surveys (CSUR)》54，3（2021），1–33。[11] Ronnie Chaiken、Bob Jenkins、Per-Åke Larson、Bill Ramsey、Darren Shakib、Simon Weaver 和 Jingren Zhou。2008。Scope：海量数据集的简易高效并行处理。《VLDB Endowment 会议录》1，2（2008），1265–1276。[12] Varun Chandola、Arindam Banerjee 和 Vipin Kumar。2009。异常检测：综述。《ACM Computing Surveys (CSUR)》41，3（2009），1–58。[13] Andrew A Cook、Göksel Mısırlı 和 Zhong Fan。2019。物联网时间序列数据的异常检测：综述。《IEEE 物联网期刊》7，7（2019），6481–6494。[14] Graham Cormode、Minos Garofalakis、Peter J Haas、Chris Jermaine 等。2011。海量数据的概要：样本、直方图、小波和草图。《数据库基础与趋势》4，1–3（2011），1–294。[15] Zahra Zamanzadeh Darban、Geoffrey I Webb、Shirui Pan、Charu C Aggarwal 和 Mahsa Salehi。2022。用于时间序列异常检测的深度学习：综述。arXiv 预印本 arXiv:2211.05244（2022）。[16] Jens Dittrich 和 Jorge-Arnulfo Quiané-Ruiz。2012。Hadoop MapReduce 中的高效大数据处理。《VLDB Endowment 会议录》5，12（2012），2014–2015。[17] Marianne Durand 和 Philippe Flajolet。2003。大基数的 LogLog 计数。载于《欧洲算法研讨会》，Springer，605–617。[18] Muhammad Fahim 和 Alberto Sillitti。2019。物联网环境中的异常检测、分析与预测技术：系统性文献综述。《IEEE Access》7（2019），81664–81681。[19] Gilberto Fernandes、Joel JPC Rodrigues、Luiz Fernando Carvalho、Jalal F Al-Muhtadi 和 Mario Lemes Proença。2019。网络异常检测综合综述。《电信系统》70（2019），447–489。[20] Jim Gray、Surajit Chaudhuri、Adam Bosworth、Andrew Layman、Don Reichart、Murali Venkatrao、Frank Pellow 和 Hamid Pirahesh。1997。数据立方体：一种关系聚合运算符，可推广分组聚合、交叉表和小计。《数据挖掘与知识发现》1，1（1997），29–53。[21] Michael Greenwald 和 Sanjeev Khanna。2001。在线计算分位数摘要的空间高效方法。《ACM SIGMOD Record》30，2（2001），58–66。[22] Songqiao Han、Xiyang Hu、Hailiang Huang、Minqi Jiang 和 Yue Zhao。2022。ADBench：异常检测基准。《神经信息处理系统进展》35（2022），32142–32159。[23] Mahmudul Hasan、Md Milon Islam、Md Ishrak Islam Zarif 和 MMA Hashem。2019。使用机器学习方法检测物联网站点中物联网传感器的攻击与异常。《物联网》7（2019），100059。[24] Paulo Jesus、Carlos Baquero 和 Paulo Sérgio Almeida。2014。分布式数据聚合算法综述。《IEEE 通信调查与教程》17，1（2014），381–404。[25] Junchen Jiang、Vyas Sekar、Henry Milner、Davis Shepherd、Ion Stoica 和 Hui Zhang。2016。{CFA}：用于优化视频{QoE}的实用预测系统。载于《第 13 届 USENIX 网络系统设计与实现研讨会（NSDI 16）》，137–150。[26] Meng Jiang、Christos Faloutsos 和 Jiawei Han。2016。CatchTartan：表示和概括动态多上下文行为。载于《第 22 届 ACM SIGKDD 知识发现与数据挖掘国际会议论文集》，945–954。[27] Dominik Kreuzberger、Niklas Kühl 和 Sebastian Hirschl。2023。机器学习运维（MLOps）：概述、定义与架构。《IEEE Access》（2023）。[28] Fei Tony Liu、Kai Ming Ting 和 Zhi-Hua Zhou。2008。孤立森林。载于《2008 年第八届 IEEE 国际数据挖掘会议》，IEEE，413–422。[29] Andreas Lohrer、Darpan Malik 和 Peer Kröger。2023。GADFormer：一种基于注意力的轨迹群体异常检测模型。arXiv 预印本 arXiv:2303.09841（2023）。[30] Antonis Manousis。2022。在多维数据流中实现高效通用的子群体分析。载于 VLDB 2022。PVLDB（2022）。[31] Md Salik Parwez、Danda B Rawat 和 Moses Garuba。2017。移动无线网络中的用户活动分析与用户异常检测的大数据分析。《IEEE 工业信息学汇刊》13，4（2017），2058–2065。[32] Nisheeth Shrivastava、Chiranjeeb Buragohain、Divyakant Agrawal 和 Subhash Suri。2004。中位数及更多：传感器网络的新型聚合技术。载于《第 2 届国际嵌入式网络传感器会议论文集》

系统。239–249。[33] Jacopo Soldani 和 Antonio Brogi。2022。基于（微）服务的云应用中的异常检测与故障根因分析：综述。*ACM Computing Surveys (CSUR)* 55，3（2022），1–39。[34] Ya Su、Youjian Zhao、Chenhao Niu、Rong Liu、Wei Sun 和 Dan Pei。2019。通过随机循环神经网络实现多变量时间序列的稳健异常检测。载于第 *25* 届 *ACM SIGKDD* 知识发现与数据挖掘国际会议论文集，2828–2837。[35] Daniel Ting。2019。数十亿数据集的近似不同值计数。载于*2019* 年国际数据管理会议论文集，69–86。[36] Panos Vassiliadis。1998。多维数据库、立方体及立方体操作的建模。载于第十届国际科学与统计数据库管理会议论文集（编号 *98TB100243*），IEEE，53–62。[37] Liang Xiong、Barnabás Póczos 和 Jeff Schneider。2011。使用灵活的类型模型进行群体异常检测。神经信息处理系统进展24（2011）。[38] Fangjin Yang、Eric Tschetter、Xavier Léauté、Nelson Ray、Gian Merlino 和 Deep Ganguli。2014。Druid：实时分析型数据存储。载于 *2014* 年 *ACM SIGMOD* 国际数据管理会议论文集，157–168。[39] Yuan Yu、Pradeep Kumar Gunda 和 Michael Isard。2009。面向数据并行计算的分布式聚合：接口与实现。载于*ACM SIGOPS* 第 *22* 届操作系统原理研讨会论文集，247–260。[40] Chao Zhang、Farouk Toumani 和 Emmanuel Gangler。2017。大规模并行计算中的对称与非对称聚合函数（扩展版）。博士学位论文。LIMOS（UMR CNRS 6158），克莱蒙奥弗涅大学，法国。

论文《AHA：面向运营时间序列应用的可扩展替代历史分析》的补充材料

AHA 对不同算法的适用性

一种可用于推导任意充分统计量的简单样本统计量，就是用户数据样本本身（即我们只需存储完整的原始数据集）。然而，由于存储成本，我们无法支持那些需要摄取所有样本的算法 [29, 37] 。因此，我们回顾了常用的样本统计量，并分析其可分解性以及对 AHA框架的适用性。

我们可以利用子组的计数和总和来计算任意父子组的均值，而这两者都是自可分解的。因此，我们可以支持计算任意子组的均值。若没有所有样本，我们就无法准确计算子组总体的中位数。不过，有可靠的近似算法可以通过存储样本直方图的频数值来进行估算 [32]。类似的方法也可用于近似估计样本分位数和百分位数。样本的方差和标准差可以根据均值、计数和平分和进行估算，这些量都是可分解的。同样，我们可以通过跟踪样本值的适当幂次之和来计算高阶矩。与中位数和四分位数类似，我们无法准确估计四分位距，但可以利用四分位数的近似估计值。由于最大值和最小值是自可分解统计量，我们可以准确估计极差。大多数标准的时间异常检测基准测试并不假设生成时间序列时所用特征的类型，或者仅使用简单统计量，例如来自一组传感器的观测值均值或总和 [15]。

B 基准设置详情

我们在标准计算环境中评估所有基线，该环境配备一颗64核英特尔至强处理器和256 GB可用主内存。每个时间步的输入数据都存储在单节点上的 Parquet 文件中。汇总数据则存储为压缩的 CSV 文件，供 AHA 使用（我们采用快速 zstd 压缩）。为了获取预测所需的相关统计量，我们会提取 CSV 文件，然后进行汇总，以评估所需群组的统计量。

B.0.1 异常检测算法：为了分析 AHA 和基线支持多种预测流程的能力，我们考虑了不同的异常检测算法。我们采用了三种在研究和实践中广泛使用的不同异常检测算法 [10, 12, 22]。首先，我们使用一个简单的基线进行评估：检测任一指标的平均值是否偏离历史均值 3个标准差（3Sigma） [34]。最后，我们采用另外两种常见的异常检测算法：k 近邻算法（KNN） [5] 和孤立森林（IsoForest） [28]。这些算法都使用跨用户观测到的指标均值作为特征来检测异常。我们没有选择复杂的神经网络算法，因为计算异常所需的时间远远超过回放系统摄取、存储和获取数据所需的时间。不过，这些算法也使用了与上述算法类似的统计量。由于三个任务的平均准确率非常相似，因此这里仅展示其平均结果。此外，三个任务的总成本中，超过 99% 都来自摄取和获取组统计量。

B.0.2 评估指标：我们通过衡量聚合指标的准确性（指标准确率）以及异常检测算法的准确性（任务准确率），来评估各基线的保真度。我们使用均方根误差分数衡量指标准确率，并通过异常的分类准确率衡量任务准确率。由于对于可分解统计量，AHA 能够完美召回所需统计量，因此我们预计指标预测和算法预测的一致性准确率均为 100%，其他具有完美召回率的解决方案也是如此：StoreRaw、StoreOutput、Key-ValueStore。由于采样和草图化都采用近似方法，这些解决方案的指标准确率和任务准确率可能会降低。我们衡量跨时间以及所有用户子组的平均准确率。我们还衡量子组间准确率的方差，因为样本量较小的子组尤其容易受到这些方法错误预测的影响。

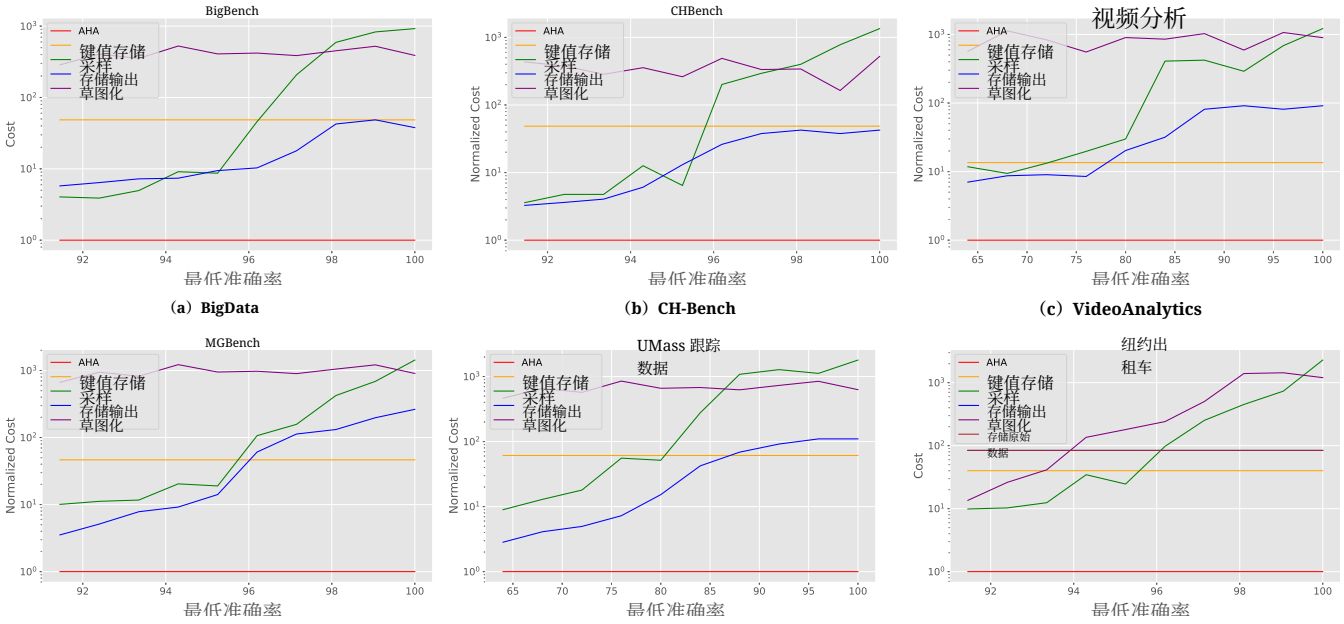
B.0.3 成本模型：为了确定系统的总成本（计算和存储），我们使用 AWS，通过 EC2 配置相近的实例，测量该操作在一个月内的执行成本；每种解决方案的汇总数据均存储在 S3 上。我们测量计算总时间，并将其乘以计算成本（每小时 0.96 美元）；同样，通过计算每 GB 的费用（每月 0.15 美元）来得出存储总成本。然后计算两者的总成本。我们报告归一化成本，即总成本除以默认的典型大数据架构解决方案 StoreRaw 的成本，例如 Spark。

B.0.4 弱等价模型的超参数校准：弱等价方法具有用于确定解决方案准确率与成本之间权衡的超参数。采样有一个超参数 k，即要存储的样本百分比。同样，草图化有一个超参数 s，即草图的大小。草图的大小等于草图中使用的哈希函数数量乘以桶的数量。我们对草图化的哈希函数数量进行调优。我们根据用户需求调优超参数，用户需求由

参数（δ，ϵ）定义，其中 δ是误差至多为 ϵ的群组百分比。默认情况下，我们设置 $\delta = 0.95$ 并针对不同的 ϵ进行调优，以确定成本权衡。

数据集	VideoAnalytics	UmassTrace	NYTaxi	CH-Bench	MGBench	BigData
属性	12	3	2	32	4	31
指标	7	1	8	11	15	7
平均组数	1303562	344	513	28711	722	119552
时间步	1081	2351	730	7116	271	1998
总大小（GB）	1667	48	2463	17.2	31.5	117.8

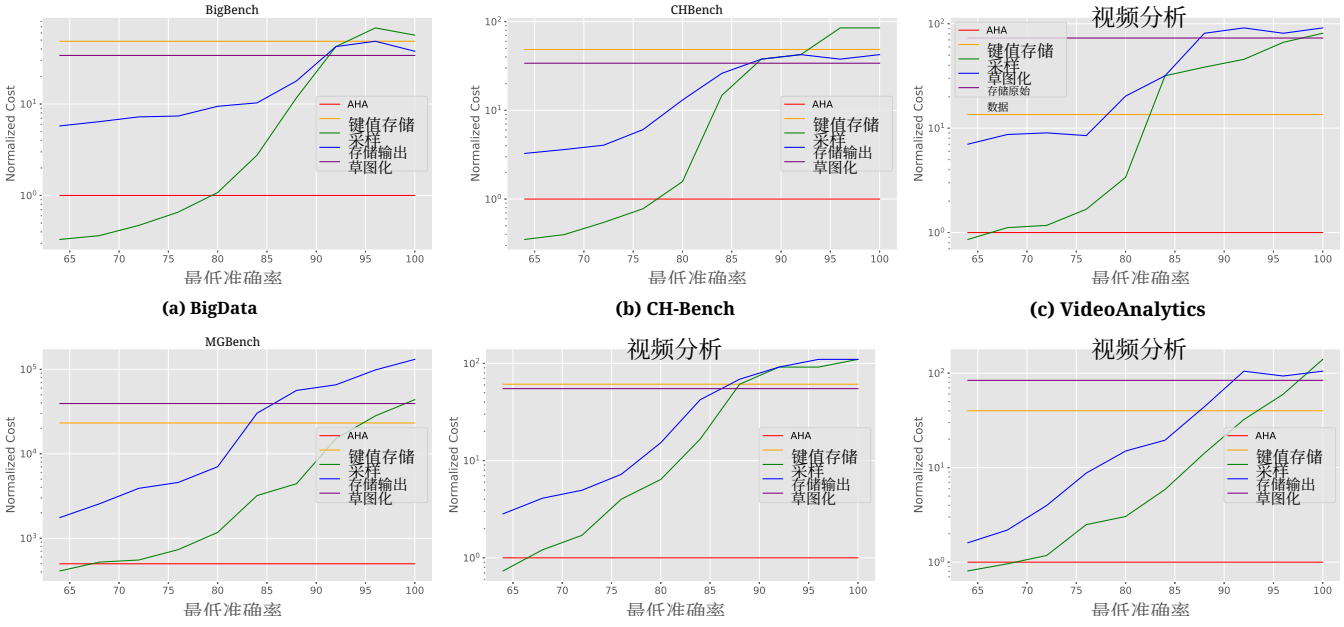
表 2：数据集统计



(a) BigData (b) CH-Bench (c) VideoAnalytics

(d) MGBench (e) UmassTrace (f) NYTaxi

图 12：不同数据集在不同最小任务准确率要求下的归一化成本。对于所有准确率要求（> 90%），AHA 都是成本最低的唯一解决方案，成本低 8.6–87 倍。另请注意，当准确率为 > 97% 时，弱等价方法的成本甚至高于存储完整会话数据的成本，这凸显了采样和草图方法在实现高准确率预测时的计算开销。



(a) BigData (b) CH-Bench (c) VideoAnalytics

(d) MGBench (e) UmassTrace (f) NYTaxi

图 13：不同数据集在不同指标准确率要求下的归一化成本。AHA 是成本最低的唯一解决方案。与强等价解决方案相比，AHA 的效率高出 55–130 倍；对于准确率要求超过 80% 的所有数据集，AHA 都是成本最低的解决方案。