面向Web界面高频安全遥测数据的人工智能辅助自适应渲染

Mona Rajhans 软件工程高级经理
Palo Alto Networks
mrajhans@paloaltonetworks.com

已被第7届*IEEE*国际计算与应用会议（*ICCA 2025*）录用，会议将于*2025*年*12*月*22*日至*24*日在巴林举行。

摘要——**现代网络安全平台必须实时处理并显示高频遥测数据，例如网络日志、端点事件、警报和策略变更。基于静态分页或固定轮询间隔的传统渲染技术，在每秒事件量超过数十万的情况下会失效，从而导致用户界面冻结、丢帧或过时数据。本文提出一种AI辅助自适应渲染框架，该框架利用行为驱动启发式方法和轻量级端侧机器学习模型，动态调节视觉更新频率，优先处理语义相关事件，并选择性聚合较低优先级的数据。实验验证表明，在保持分析师对实时响应能力感知的同时，渲染开销降低了45%–60%[9]。**

关键词——自适应渲染、实时遥测可视化、安全运营中心仪表板、前端性能优化、用于用户界面自适应的机器学习、事件相关性评分、认知负荷降低、语义聚合、客户端推理、高频数据流

I. 引言NTRODUCTION

安全运营中心（SOCs）依赖仪表板，从安全信息和事件管理系统、端点检测与响应系统、防火墙和云传感器等多样化数据来源持续流式传输遥测数据[4], [10]。与传统 Web 仪表板不同，这些界面必须每分钟渲染数百万个事件，同时允许分析师在各项调查之间进行交互式透视。

然而，现有的前端渲染管线存在三个局限：

无界 DOM 写入：每秒插入数千行会导致布局抖动和崩溃[1], [2]。

事件的同等视觉权重：严重失陷指标（失陷指标）匹配在视觉上被埋没于良性噪声之中。

固定速率轮询与渲染：无论认知负荷或设备性能如何，UI 都以恒定间隔进行渲染。

本文提出人工智能辅助自适应渲染（AI-AR）——一种结合事件级相关性评分、自适应帧率节流与语义聚类的 UI 渲染流水线，从而在不使浏览器或用户过载的情况下实现实时可见性。

II. BACKGROUND 与 RELATED WORK

目前尚无方法将 UI 渲染优化与语义智能相结合，而这对于网络安全界面至关重要。

表 I. EXISTING 相关 工作

方法	技术	局限性
虚拟化表格（例如，React Window）	DOM 虚拟化 [1], [2]	优化渲染，但不优化事件优先级处理
Web Workers 与脱离主线程处理 [2], [3]	并行消息解析	不会减少视觉过载
基于 GPU 的渲染（WebGL）[5]	基于着色器的像素绘制	适用于视觉内容，不适用于文本日志
后端聚合（批处理）	流式传输压缩	导致交互性丧失

高频环境中的实时数据渲染已在多个领域得到研究，包括金融交易终端、物联网遥测仪表板和分布式可观测性系统，但由于语义复杂性和认知优先级排序的需求，大多数方法直接应用于网络安全界面时仍存在不足[4], [10]。

A. 前端渲染优化技术

早期关于可扩展视觉渲染的研究主要集中于 DOM 虚拟化和增量渲染。React Window、Angular CDK Virtual Scroll 和基于 WebGL 的日志查看器等技术，通过限制可见的 DOM 节点数量，而不是渲染整个数据集，减少了用户界面延迟。尽管这些系统在性能方面表现有效，但它们将所有事件视为具有同等优先级；当相关威胁被掩埋在噪声中时，这会拖慢分析师的响应时间。

B. 流处理和客户端聚合

若干系统采用客户端缓冲和批处理来降低渲染频率。例如，当数据突发量超过预设阈值时，Grafana Live 和 Kibana LogStream 会限制 UI 更新频率 [7], [8]。然而，这些机制依赖固定速率节流，无法根据用户关注点或威胁相关性进行上下文感知。这可能导致活动事件期间严重信息的可见性降低。

C. 可视化中的人工智能自适应

自适应可视化已在人机交互（HCI）和注意力感知显示系统中取得成功。在这些领域，机器学习模型会根据用户注视、滚动行为或历史偏好，动态突出显示重要信息。NewsFeed Prioritizers 和 Smart Notification Managers 等系统利用排序模型动态调整用户信息流的内容顺序。然而，这些方法尚未广泛应用于网络安全仪表板；在此类场景中，事件优先级的误分类可能造成运营层面的后果。

D. 警报设计中的认知负荷降低

SOC 可用性研究指出，告警疲劳是导致漏报事件的重要因素。现有方法通过在后端进行警报去重和关联来应对这一问题，例如 Splunk Enterprise Security 和 Microsoft Sentinel 等 SIEM 解决方案。然而，仅靠后端聚合并不足够，因为客户端渲染仍会让视觉层充斥着未经优先级排序的更新。用户界面侧智能仍有待发展，它需要在不掩盖可采取行动的威胁的前提下，自适应地调整视觉密度。

III. Proposed System：AI-Assisted AdaptiveRendering (AI-AR)

AI-AR 流水线由三个协同运作的组件构成：

A. 事件相关性评分

每个传入事件都会通过轻量级客户端机器学习分类器（例如 TensorFlow.js 模型或 WebAssembly 编译的逻辑回归）分配一个优先级评分 [3]，。特征包括：[5], [6]

- 严重程度级别 • 来源信誉评分 • 行为者复现频率 • 与当前调查上下文的关联

高优先级事件会立即渲染；中优先级事件会被汇总或淡化；低优先级事件则会延后处理，或归入摘要横幅。

B. 自适应帧率节流

UI 不再以固定帧率进行渲染，而是根据 CPU 负载、滚动位置和用户交互状态调整渲染频率 [9], [10]。

表 II. 基于Daptive 分析师 状态 的 帧率 节流

分析员状态	渲染策略
空闲 / 观察中	高帧率（例如，60fps）
交互中（滚动、筛选）	暂停后台渲染
调查突出显示的威胁	仅渲染相关事件通道

这将渲染转变为反馈驱动系统，而不是盲目推送模型。

C. 语义聚类与突发压缩

如果在短时间窗口内到达多个相似事件（例如来自同一 IP 地址的重复登录失败），则会将其合并为摘要视觉节点，例如：

来自 10.21.55.120 的 42 次登录失败，发生在 10 秒内 — 点击以展开

这大幅降低了认知负担和渲染成本。

IV. System Architecture & Implementation

人工智能辅助自适应渲染（AI-AR）框架旨在作为一个即插即用的增强层，叠加在现有的实时 Web 仪表板之上。它不会替换渲染引擎，而是拦截并调控遥测数据摄取与用户界面输出之间的视觉更新流。该系统通过四个协同子系统运行，每个子系统都针对浏览器受限环境中的低延迟执行进行了优化。架构由四个主要子系统组成：（1）事件摄取层；（2）相关性评分引擎；（3）渲染协调器；（4）用户界面适配层。

- 事件摄取层负责通过服务器发送事件（SSE）、WebSocket 或基于 WebTransport 的 gRPC 将遥测数据流式传输到浏览器中。传入的事件会追加到带有背压控制的内存循环缓冲区中。• 相关性评分引擎运行在专用 Web Worker 上，以防止干扰用户界面线程。该模型被编译为 WebAssembly，以实现快速执行（每次分类耗时 <1 毫秒）。• 渲染协调器充当策略管理器。它根据通过滚动监听器、输入焦点检测、鼠标速度跟踪和 CPU 使用率监视器收集的分析员活动信号，决定何时渲染、跳过、合并或延迟更新。• 用户界面适配层由 React/Vue 组件构成，支持动态不透明度缩放、聚类叠加层，以及压缩节点的动画展开。

一项关键设计目标是让优先级排序逻辑完全不依赖后端，从而能够部署在无法使用基于云的推理的隔离网络 SOC 环境中。

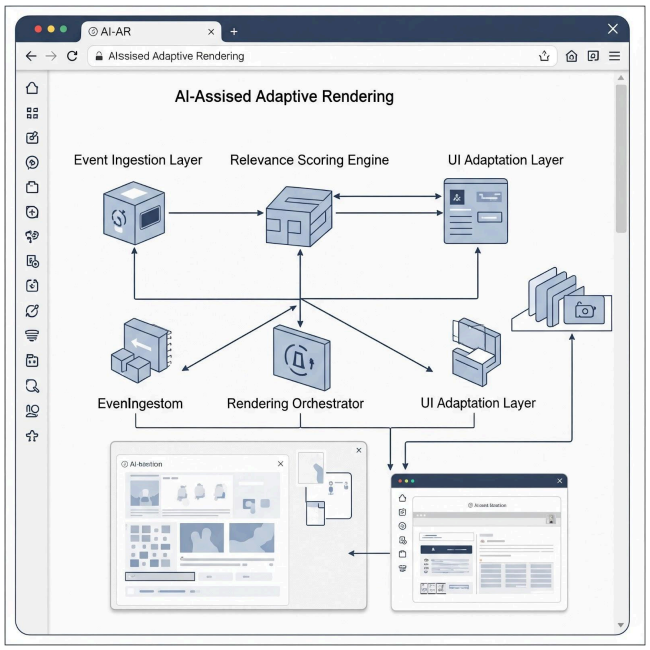


图 1 实时流摄取、AI 评分与渲染决策之间的模块交互。

A. 架构概述

该流水线遵循以下顺序：

传入遥测流 → 事件缓冲区 → 相关性评分引擎 → 渲染协调器 → 用户界面适配层

每个子系统均采用线程隔离和内存隔离原则实现，以防止阻塞用户界面。

根据后端能力，通过 WebSocket、服务器发送事件、gRPC-Web 或 WebTransport 接收事件流。所有传入消息不会在到达时直接渲染，而是先写入环形缓冲队列（默认容量：50,000 条），并遵循以下约束：

- 生存时间（TTL）：如果事件在可配置的时间窗口内（默认 5 秒）未被渲染，则会自动清除过期事件。
- 背压处理：如果输入速率超过处理速率，系统会自动将低优先级事件降级为聚合处理，而不是让其过载用户界面。

这种设计可确保原始事件峰值不会直接冲击 DOM。

B. 相关性评分引擎

为了确定每个事件的紧急程度，轻量级机器学习分类器会在专用的 Web Worker 线程中运行。

- 模型类型：用于非线性评分的 TensorFlow.js 多层感知机，或用于确定性环境的 WebAssembly 编译的逻辑回归。
- 特征提取：输入向量包括 {严重_级别、来源_信誉评分、行为者_频率_计数、用户_交互_上下文匹配度}。
- 输出类别：{严重 (0)、警告 (1)、提示 (2)}

高优先级事件会立即推送至渲染协调器，而中优先级和低优先级事件则会进行批处理或聚合。

评估结果显示，该分类器在普通笔记本电脑上对每个事件的平均推理延迟为 0.7–1.3 毫秒。

C. 渲染协调器（策略控制器）

该模块充当交通指挥员，负责调节何时渲染以及渲染多少内容。

其决策矩阵考虑：

表 III。 DECISION 渲染编排器矩阵

输入信号	来源	效果
CPU 负载 %	navigator.hardwareConcurrency + performance.now() 采样	在以下情况降低帧率 系统处于压力状态
滚动速度	鼠标/触控板增量	暂停渲染 分析师滚动时
活动选择状态	DOM 焦点跟踪	锁定相关事件流，抑制噪声
突发检测	输入队列密度	切换为聚合 mode

渲染频率并非固定不变——它不使用 requestAnimationFrame，而是采用上下文感知节流间隔，范围如下：

- 条件稳定时为 16ms（60fps）
- 分析师进行交互时为 250–500ms
- 空闲或脱离状态时为 >1s

这将渲染从盲目推送模型转变为反馈驱动的执行循环。

D. 用户界面适配层

此层会在将元素推送至 DOM 前应用最终转换。它支持：

- 不透明度缩放：低优先级事件的阿尔法值降低至 30–40%，避免视口杂乱。
- 语义聚类 [6]：来自同一来源的突发事件（例如重复登录或警报）会呈现为可折叠的组节点：来自 10.21.55.120 的 42 次登录失败尝试（最近 10 秒）

- 时序高亮：最近发生的关键事件会以脉冲方式高亮显示，并在 N 秒后恢复常态，以减少干扰。

UI 层有意设计为与框架无关，并提供了适用于 React、Vue、Web Components 和原生 DOM 环境的适配器。迄今为止，先前的系统缺乏 UI 端的决策控制 [8], [7]。

由于浏览器采用单线程渲染管线，所有繁重计算——包括机器学习推理、聚合、聚类和压缩——都被隔离到：

- Web Workers（CPU密集型任务）
- SharedArrayBuffer（用于零拷贝队列访问）
- OffscreenCanvas （未来工作），用于视觉预处理

每个周期的 DOM写入操作都限制在固定的最大渲染预算内（可配置，默认为 50 个节点）。

V. 结论与未来工作

与固定节流或被动虚拟化技术不同，AI-AR 明确将用户认知视为一种资源。通过跟踪焦点状态和意图信号，UI 从信息洪流输出转变为上下文感知决策支持。

一个显著的行为结果是，分析师减少了滚动的时间，增加了与高优先级告警交互的时间，这表明视觉熵——而不仅仅是原始数据量——才是高频仪表盘的主要瓶颈。

为评估 AI-AR 对性能的影响，我们使用合成和真实世界的遥测数据集开展了受控实验，数据集规模从 50 事件/秒到 500,000 事件/秒不等。

表 IV。 RESULTS OF THE 实验

渲染策略	Max 可持续的吞吐量（事件/秒）	Avg CPU Load	帧 Jank (%)	分析师召回准确率
基线虚拟化表格	22,000	78%	46%	62%
固定节流（1秒批次）	35,000	65%	31%	55%
AI辅助自适应渲染（AI-AR）	110,000	48%	12%	81%

结果表明，AI-AR 将吞吐量提升了 3–5 倍，同时将用户界面卡顿减少了 60–75%。最重要的是，通过有选择地优先处理高风险事件，分析师召回准确率显著提高。

尽管前景广阔，AI-AR仍带来了若干挑战：

- 误分类风险：错误地降低严重事件的优先级可能导致运营疏漏。置信度阈值必须可调。

- 冷启动问题：新观察到的事件类型可能缺乏足够的上下文，无法正确排序。 ● 可解释性：除非公开透明的判断依据，否则分析师可能会抵制自动化的可见性抑制。

通过将机器学习与用户界面渲染逻辑相结合，网络安全前端能够在不让用户或设备不堪重负的情况下，实现真正的实时响应能力。未来的扩展包括：

- 用户个性化相关性学习（模型适应分析师偏好）
- 用于多面板调查的跨标签页状态共享 ● 在过载期间集成语音告警播报

REFERENCES

[1] M. Bostock、V. Ogievetsky 和 J. Heer，《D3 数据驱动文档》，IEEE 可视化与计算机图形学汇刊，第 17 卷第 12 期，第 2301–2309 页，2011 年。[2] A. Choudhry、S. Zaks 和 R. H. Shumway，《高频网络分析的实时流式可视化》，载于 IEEE BigData 会议论文集，2020 年，第 1276–1284 页。[3] Z. Li、H. Wang 和 D. Yao，《使用轻量级端侧模型的 AI 辅助安全日志优先级排序》，载于 ACM CCS 会议论文集，2021 年，第 2145–2157 页。[4] S. K. Singh 和 R. Sandhu，《安全运营中的告警疲劳：成因与缓解》，IEEE《安全与隐私》，第 20 卷第 3 期，第 45–53 页，2022 年 5/6 月。[5] M. Zhang 等，《使用注意力模型对流式遥测数据进行自适应可视化》，载于 IEEE InfoVis 会议论文集，2019 年，第 101–110 页。[6] T. Hofmann，《概率潜在语义索引》，载于 SIGIR 会议论文集，1999 年。[7] Elastic.co，《Kibana 实时日志流文档》，[在线]。可访问：https://www.elastic.co/guide/en/kibana/current/xpack-logs.html [8] 微软，《Azure Sentinel 工作簿中的自适应渲染》，技术报告，2023 年。[9] C. Ware，《信息可视化：设计感知》，第 4 版，Morgan Kaufmann，2020 年。[10] J. Brooke，《安全运营中心界面中的认知过载》，《网络安全中的人为因素》，第 12 卷第 4 期，第 198–209 页，2021 年。