ETCpipe: Streamlining Encrypted Traffic Classification with Advanced DPU/GPU Integration

Lingxiang Hu

Abstract

随着网络流量模式的日益复杂和加密通信技术的广泛应用，基于深度学习的实时流量分析需求急剧增加。尽管新兴的深度学习网络模型在加密流量分类上表现出色，但在现有服务器上执行这些任务时对资源的消耗巨大。目前的解决方案，如 FENXI 系统，尽管提升了性能，但仍面临配置复杂性的问题，且依赖于统计特征而忽略了基于原始负载数据的分析，限制了系统的通用性。

本研究引入了“ETCpipe”，一个针对 ETC 任务设计的执行系统，通过将流量管理和特征提取任务解耦至领域特定加速器，显著释放了主机 CPU 资源，并消除了配置问题。我们创新性地将流量管理任务卸载至 SmartNIC 的 FlowManager，同时，利用 GPU 上的特征提取组件 GPUFE，并借助推理引擎完成分析任务。面临的挑战包括 GPU 处理过滤后数据包的流重组难题，我们通过 GPUdirect 技术和并行算法解决，直接将数据包传递至显存中，优化了特征提取过程。

我们在 V100 GPU 和 BlueField-3 DPU 平台上实现的 ETCpipe 系统原型证明了其高效性，通过端到端的性能测试显示，ETCpipe 对网络吞吐量没有影响，且在特征提取性能上达到了传统 CPU 解决方案的三至四倍。本研究的主要贡献包括系统设计的深度解耦和对原始数据包负载模型的深入分析，这不仅简化了配置复杂性，还提高了系统的处理速度和通用性，为实时流量分析技术的进步提供了新的方向。

Introduction

在现代网络架构中，实时流量分析是不可或缺的一环，面对日益增加的网络流量模式复杂性和加密通信技术的广泛采用，基于深度学习（DL）的流量分析方法需求持续增长。尤其在加密流量分类（ETC）上，新兴的深度学习网络模型已展现出卓越性能，成为处理关键网络流量分析任务的有力工具。然而，这些深度学习任务在现有服务器上执行时，对资源的消耗巨大，涵盖了流量管理、特征提取（或称为数据预处理）和模型推理等多个环节。网络速度的显著提升——超过 100 Gbps——要求实时推理系统不仅要在保持网络通信流畅的同时，还必须及时处理和分析大量流量。这一复合要求对系统的处理能力和资源分配策略提出了双重挑战，迫切需要解决方案以维持任务的高效运行。

面对实时分析系统的挑战，业界已提出多种解决方案，其中 FENXI 代表了当前最先进的技术（State-of-the-Art, SOTA）。FENXI 通过解耦网络数据平面与流量分析任务并构建高效的处理流水线，显著提升性能。该方案利用多核 CPU 进行流量管理和特征提取，同时采用 GPU 或 TPU 执行深度学习模型的推理。尽管如此，FENXI 面临着显著的配置复杂性问题：流水线的资源分配需人工调整，要求管理员对硬件性能与模型需求有深入理解，以便优化系统性能。此外，FENXI 和其他现有方法普遍依赖于统计特征（如包长度和到达时间），却忽略了基于原始负载数据的深入模型分析，这一做法限制了系统的通用性和对复杂网络环境的适应性。

在此研究中，我们引入了“ETCpipe”，一个专为异构 ETC 任务设计的执行系统。本系统

的核心创新在于，它将流量管理和特征提取任务进一步解耦至领域特定加速器，从而释放了主机 CPU 资源，消除了配置问题，并实现了更快的处理速度。具体来说，我们将流量管理任务卸载至 SmartNIC，称为 FlowManager。FlowManager 负责过滤流量，筛选出需分析的数据包并传递给 GPU。随后，GPU 上的特征提取组件，称为 GPUFE，提取流量的特征信息，最终借助推理引擎（如针对 NVIDIA 显卡的 TensorRT）完成分析任务。

该解耦带来了显著挑战：虽然 GPU 以其高度并行的能力在收集足够数量的数据包后进行特征提取和推理看似是合理的解决方案，但在处理过滤后的数据包时，GPU 难以直接完成流重组任务。这要求 CPU 介入进行流重组，并再将数据拷贝至显存中，未能完全卸载流量管理任务，且增加了一次内存拷贝过程。

为克服这些挑战，我们采用了 GPUdirect 技术，直接将过滤的数据包从网卡传递至显存中而不经过 CPU 主存，并通过排序和前缀扫描等算法实现流重组，从而并行地提取特征。我们在 V100 GPU 和 BlueField-3 DPU 平台上实现了 ETCpipe 的系统原型，并对 FlowManager 和 GPUFE 两个关键组件进行了测试。通过 iperf 工具进行的端到端测试显示，相比于传统的 DPU ovs，ETCpipe 的高度并行设计使其对网络吞吐量几乎无影响。同时，使用 pktgen-dpdk 生成的数据包对 GPUFE 组件进行特征提取测试表明，仅使用一个 1024 线程块的 GPUFE 组件就能达到 FENXI CPU 解决方案的三至四倍性能。

本研究的主要贡献可以概括为以下几点：

（1）系统设计的创新：本研究首次实现了将 CPU 承载的任务深度解耦至领域特定加速器的系统设计，显著简化了当前系统中存在的配置复杂性。这一设计不仅优化了资源利用效率，还提高了整个系统的处理速度。

（2）原始数据包负载模型的分析：我们提出的框架是首个针对基于原始数据包负载进行深入分析的系统，有效地填补了现有研究中对此类分析缺乏的不足。通过对原始负载数据的直接分析，我们的框架能够提供更精准的流量分类和安全分析，从而增强了网络监控和管理的能力。

Related Work

本节回顾了深度学习在加密流量分类（Encrypted Traffic Classification, ETC）领域的应用，并简要介绍了加速这一过程的关键硬件技术：数据处理单元（DPU）和图形处理单元（GPU）。

2.1 深度学习在 ETC 的应用

加密流量分类（ETC）领域根据输入特征不同，分为基于数据包的原始信息和原始有效载荷两种。前者能够基于数据包的原始信息（如数据包大小、到达时间以及数据包内容），例如，研究工作 DF 采用卷积神经网络（CNN）从加密流量的数据包大小序列作为输入；FS-Net 利用循环神经网络（RNN）进行相似的任务；后者直接以原始有效载荷作为输入特征，例如 Deeppacket 和 TSCRNN。最近的研究，如 LLM 和 ET-BERT，也在 ETC 任务中证明了深度学习模型的出色性能。

2.2 加速器

在构建高效的异构加速流水线以优化加密流量分类（ETC）任务的过程中，对不同硬件加速器的理解至关重要。本节详细介绍了两种核心加速器：数据处理单元（DPU）和图形处理单元（GPU），它们在提升 ETC 任务处理效率中扮演着关键角色。

2.2.1 DPU

数据处理单元（DPU）是专为网络数据处理而设计的高度专业化硬件。它融合了可编程逻辑和专用加速器，如 ARM 处理核心及针对特定流量处理任务定制的领域特定加速

器，实现了在一个集成芯片上的多功能集成。DPU 提供了一系列丰富的接口，支持开发者针对网络功能进行高效的加速处理。例如，NVIDIA 的 BlueField-3 DPU 通过集成额外的 ARM 工作核心和高度可配置的 eSwitch Match-action 表，支持执行复杂的逻辑操作和高效率的数据转发，使其成为优化网络流量管理任务的理想选择。如图 X 所示

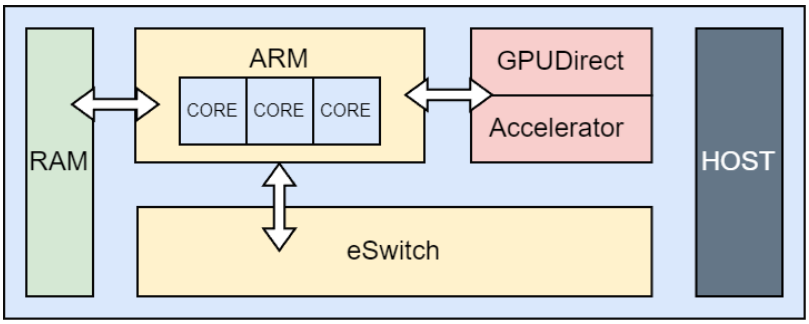

图 Bluefield-3 DPU Architectures

2.2.2 图形处理单元（GPU）

图形处理单元（GPU）以其众多计算核心在并行计算领域的卓越性能而著称，已经成为深度学习、图像处理和高性能计算（HPC）任务加速的关键技术。GPU 通过并行处理大量数据，显著缩短了复杂计算任务的执行时间。

Motivation

3.1 FENXI

在我们的研究领域内，FENXI 系统是实时加密流量分类（ETC）处理的一个重要工作。该系统采用 CPU 执行流量管理和特征提取任务，并利用 TPU/GPU 加速模型推理过程。FENXI 通过解耦数据包转发与模型分析任务，从而避免了模型推理过程可能导致的数据包转发延迟。然而如下图所示，FENXI 需要配置流水线，原文中的配置是 1:1:1，并在其研究的 case 上展示了良好的性能。

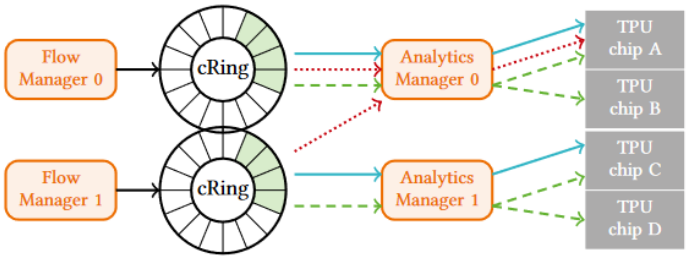

图 FENXI 配置流水线

我们对更多的 ETC case 进行本地推理，如图 X 所示，我们可以显而易见的观察到，由于不同的模型采用不同的深度学习结构以及参数规模的不同，不同的模型运行所花费的时间差距巨大；另一方面推理平台对推理的时间也有巨大的影响，比如在 pytorch 框架下，cpu 和 gpu 的推理速度差距巨大。因此，采用 FENXI 的分析框架就需要管理人员根据了解模型的结构和参数大小，同时也要了解推理的硬件平台和软件框架，否者可能由于错误的配置导致特征提取或推理的任意一方成为阻碍系统性能的瓶颈。举一个具体的例子：对于 Darknet 模型，由于推理执行时间很短，可能需要 2-3 个 cpu 核心用于特征的提取；对于 TSCRNN 模型，由于推理执行时间较长，如果一直配置 2-3 个 CPU 核心就会导致

CPU 空转和资源的浪费。

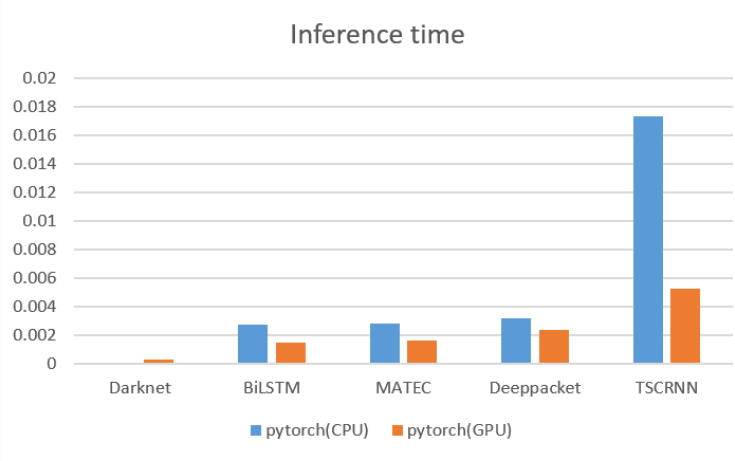

图 不同模型在不同平台的执行推理

3.2 基于原始数据包负载的特征

在现有的加密流量分类（ETC）系统中，研究主要集中于依赖统计特征的模型，例如数据包的长度和到达时间，而未充分考虑基于原始负载数据的分析。这种做法限制了模型在处理原始负载数据时的适用性，从而限制了系统的应用场景。原始负载数据作为 ETC 任务的关键输入特征之一，其分析对于理解加密流量的性质至关重要。因此，不足的通用性对现有系统提出改进需求。

Systems Design

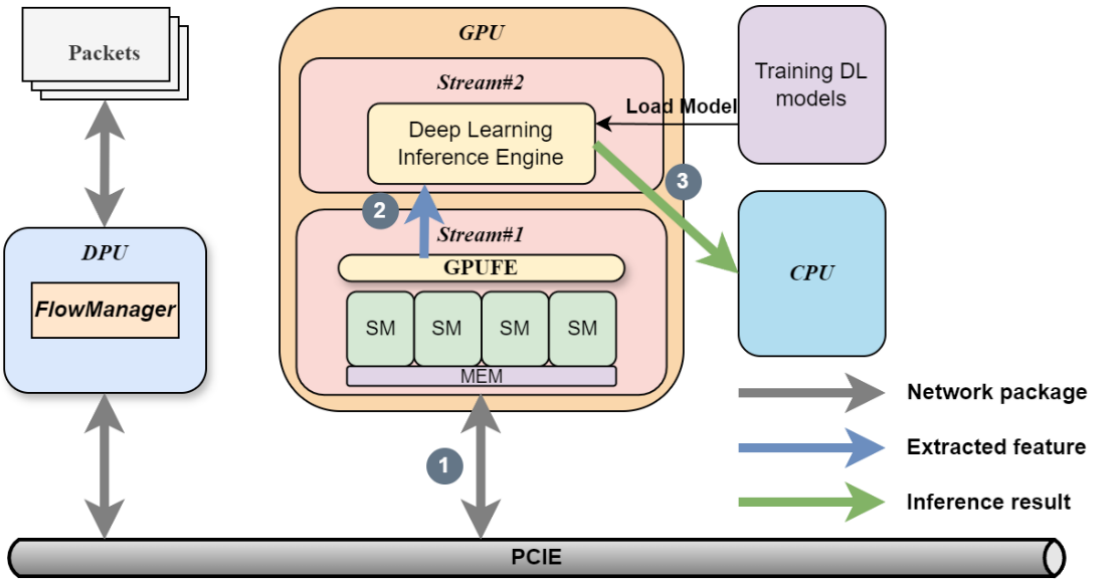

图 ETCpipe 系统架构

本节根据动机给出 ETCpipe 的设计目标，并提出系统框架与流程。

4.1 设计目标

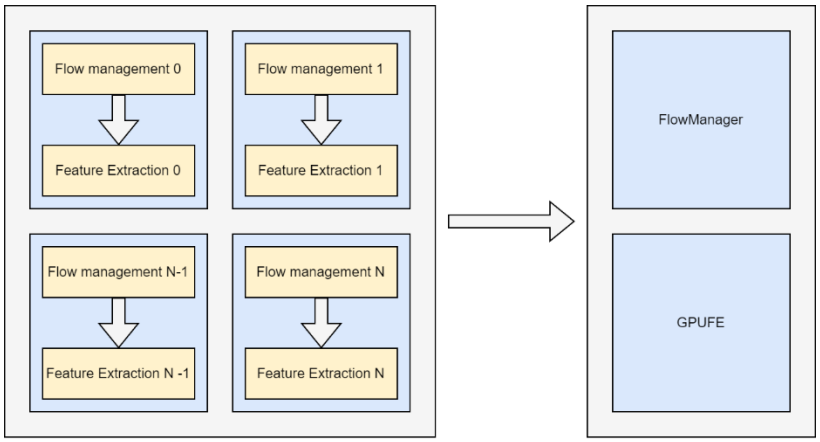

图 关键见解

本研究致力于彻底解耦流量管理与特征提取任务，关键的见解如图所示，将原有的需要配置的任务解耦，并在硬件加速器上实现这一目标，同时结合先进的推理框架来部署深度学习驱动的加密流量分类（ETC）分析系统。我们提出的方案旨在简化网络管理员在部署 ETC 任务时的配置流程，具体设计目标如下：

快速特征提取：我们追求在特征提取过程中实现计算优化和数据传输的效率。在算法层面，我们计划开发优化的计算方法，采用并行处理策略以加速数据处理流程。在数据架构方面，目标是最小化内存及异构硬件间的数据传输，降低延迟和带宽占用，实现更高效的特征构建。

全面的特征分析：鉴于现有系统多依赖于数据包大小和到达时间等统计特征，我们计划将分析扩展至基于原始数据包负载，提升系统对各类 ETC 应用的适用性。这样的全面特征分析旨在填补现有系统在原始数据处理方面的空白，确保即便在推理设备升级后，系统配置依然稳定，减少需要管理员干预的频率。

高吞吐性能：在数据中心边缘的 ETC 部署应增强服务质量（QoS）和网络安全性。我们设计的系统将确保数据包在整个模型分析过程中的高吞吐率和低延迟，以防止任何对网络服务质量或用户体验的不良影响。这意味着，不仅推理分析必须高效，而且数据包的处理和转发也需要符合网络性能的严格标准。

4.2 系统架构概述

本节阐述了 ETCpipe 系统的设计理念和工作流程。如图所示，ETCpipe 系统架构如下所述，解耦了流量管理和特征提取任务至专用的硬件加速器。流量管理在数据处理单元（DPU）上进行，特征提取则交由图形处理单元（GPU）负责。系统由三个主要组件构成：流量管理器（FlowManager）、特征提取器（GPUFE）以及推理引擎。

流量管理器：我们将该组件命名为 FLowManager，该组件部署在 DPU，负责接收并按照特定条件过滤数据包。基于网卡多队列 RSS，启动多个 ARM 核，每个核创建一个 hash 桶，根据数据包的五元组信息，ETCpipe 能够识别并追踪流，如果是符合需求的数据包，FLowManager 将复制一份放入 RING 队列并将原版的继续转发，该流程有效解耦了数据分析任务与数据包转发过程，确保了系统的高吞吐。定期刷新 RING 队列，将符合要求的数据包转发到特征提取器。

特征提取器：我们将该组件命名为 GPUFE，该组件的部署在 GPU 上，主要职责是接收过滤后的数据包并并行地执行特征提取任务。通过采用 GPU Direct RDMA 技术，组件能够直接将流管理器中的数据包传输至 GPU 显存中。接着，组件通过高效流重组算法，获得所

需的流输入特征。我们将在具体实现的章节，解释为什么需要在 GPU 上做流重组以及算法的实现。

推理引擎：推理组件负责推理提取的特征，它可以是基于 CPU、GPU 甚至是 TPU。在本文的上下文场景中，由于原型实现基于 V100，选择基于 Nvidia GPU 的 TensorRT 是一个合适的选择。特征提取组件完成特征处理后无需额外的内存拷贝操作即可直接进行推理。

综上所述当数据包进入系统时，首先，经过 FLowManager，如果不满足需求则直接转发，满足条件则放入 FLowManager 的 RING 队列并定期传递给 GPUFE，之后 GPUFE 进行流重组提取特征，最终满足要求的特征将传入系统的推理引擎进行推理得到结果。通过这样的解耦设计，管理人员只需要在意要提取哪些数据包（在流量管理中指定过滤的条件），而无需配置流水线。

5. FlowManager

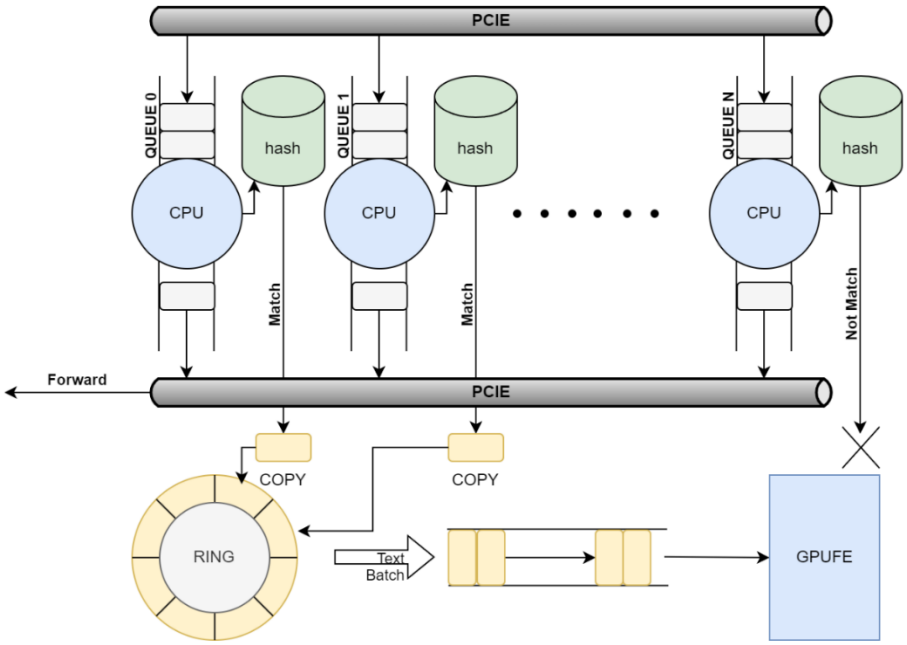

图 FlowManager 框架

FlowManager 组件是 ETCpipe 系统处理的起点，负责确保网络通信的无缝和高效率。在执行流量管理的同时，它对进入系统的流量进行状态跟踪和必要的过滤，为 GPUFE 组件的特征提取任务做前置准备。图 X 展示了流量到达 FlowManager 的初始处理步骤：数据包首先被均匀分配至多个处理队列，然后根据预定义的过滤规则进行筛选。符合条件的数据包被复制到 DPU 上的 RING 队列，并以批量方式传输到 GPUFE。本节将详细阐述 FlowManager 的设计和实现细节。

FlowManager 的核心任务是实时流量状态跟踪。为适应各种 ETC 任务的输入需求多样性，例如，某些任务可能只需要流的前 N 个非 ACK-FIN 数据包，FlowManager 利用 DPU 提供的广泛计算资源（包括通用计算资源和专用硬件加速器）进行高效的处理。它通过部署多个 ARM 核心，构建了一个高效的多处理队列体系。每个 ARM 核心分别管理一个入队列和出队列，并维护一个以五元组为键、流状态为值的哈希桶。网络接口卡（NIC）的接收端口规模集（RSS）功能根据数据包的五元组信息计算哈希值，并将数据包均匀分散到不同处理队列。这确保数据包分布的均衡性和处理的并行性，显著减少了单个处理队列的负载并提升了整体吞吐能力。

最终，这一策略使 FlowManager 能够支撑高速的流量分析，同时保持网络通信的低延迟和高可靠性。FlowManager 的性能和效率将在第 7 章的端到端测试中进行详细评估，以验证其在实际网络环境中的表现和对整个 ETCpipe 系统性能的贡献。

6. GPUFE

在 ETCpipe 系统中，特征提取环节是将原始网络流量转换为深度学习模型可解析的输入特征的关键步骤。继 FlowManager 完成对数据流的初步筛选与识别后，GPUFE 模块负责进一步对这些数据进行预处理，并转换为模型可用的输入特征，从而为推理过程提供必要的数据输入。为了充分展示 ETCpipe 系统如何填补了依赖原始数据包负载进行特征提取的研究空白，本章节将专注于基于原始数据包负载的特征提取过程，探讨传统基于 CPU 的特征提取实现的局限性、采用 GPU 加速所面临的挑战，以及 GPUFE 模块如何有效解决这些问题。

6.1 基于 CPU 的特征提取

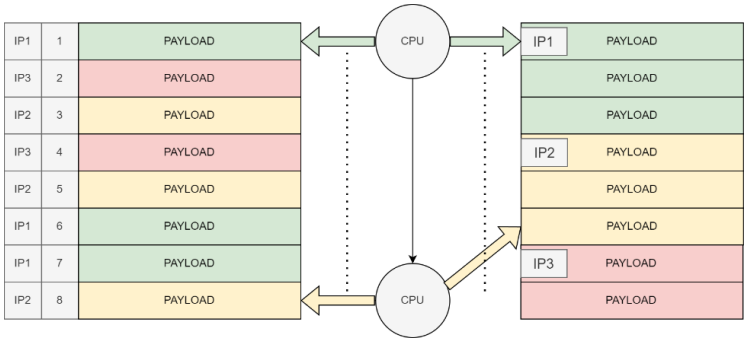

图 基于 CPU 的特征提取（着重展示基于原始数据负载）

在之前的工作中，特征提取通常通过 CPU 完成。如图展示了一个典型的基于 CPU 的特征提取过程，其中以 8 个数据包为例。在这一过程中，CPU 需逐一处理接收到的数据包。由于数据包是按不同流穿插到达的，CPU 必须从第一个数据包开始，串行地识别各个流并提取特征。一旦某个流的特征提取完成，该流的特征便被送往后续的推理模块。尽管部署多个 CPU 核心能在一定程度上提升处理性能，但如前所述，设计有效的流水线配置对于系统管理员而言是一大挑战。这不仅要求管理员对深度学习模型的需求和硬件特性有深入的理解，还需要他们综合考虑以最大化资源利用率。

此外，基于 CPU 的特征提取方法在处理高速网络流量时面临着显著的性能瓶颈。数据包的串行处理机制限制了处理速度，尤其是在数据流量激增的情况下，需要分配更多的 CPU 资源用于特征提取，这进一步增加了系统配置的复杂性。

综上所述，尽管基于 CPU 的特征提取方法在早期的 ETC 系统中得到了广泛应用，但其在处理效率、资源利用和系统配置方面的限制，使得寻求更高效的特征提取方案成为 ETC 领域的迫切需求

6.2GPUFE

6.1 技术挑战及解决策略

在探索 GPU 加速特征提取过程中，我们面临了一系列技术挑战，尤其是关于流重组和资源管理的问题。以下是详细分析及针对性解决方案。

流重组的挑战：特征提取任务的核心是流的识别与重组，如图所示，这一过程要求精确地根据流的顺序进行数据重组和特征提取。尽管基于 CPU 的处理策略可以通过串行处理实现流重组，但此方法难以直接迁移到 GPU 上，因为直接采用与 CPU 相同的算法会引入两个

主要问题。首先，为了保证数据包的顺序处理，需要在 GPU 上实施锁机制，这将大幅降低 GPU 的并行处理能力。其次，从 CPU 向 GPU 拷贝数据包也会显著影响性能，由于数据传输的开销。

资源抢占的挑战：特征提取是系统中的关键组件，但是推理引擎可能使用 GPU 资源导致竞争。从特征提取的性能角度考虑，虽然使用持久化 kernel 和更多的线程块会提升特征提取效率，但这样做可能会导致推理引擎在访问 GPU 资源时遇到竞争，影响系统的整体性能。因此，设计时必须在保证足够性能的同时，合理分配 GPU 资源，避免过度占用。

解决策略：为应对上述挑战，我们采取了以下措施：

资源限制：通过对 GPU 资源的精细管理，当检测到有 1024 个数据包到达时，我们启动一个包含 1024 线程的线程块来处理这些数据包。这种方式既充分利用了 GPU 的并行处理能力，又避免了资源的过度占用。

GPU Direct 技术：利用 GPU Direct 技术，实现了数据包从网络接口卡（NIC）直接传输到 GPU 内存的过程，减少了 CPU 到 GPU 的数据拷贝需求。这一策略显著降低了数据传输的开销，为高效的并行算法实施提供了基础。

并行算法设计：在 GPU 上，我们开发了专门的并行算法来实现流重组。这一算法针对 GPU 的计算模型优化，最大化了流重组和特征提取过程的效率。我们在 6.2 详细的讨论算法的设计与实现。

6.2GPUFE 组件实现

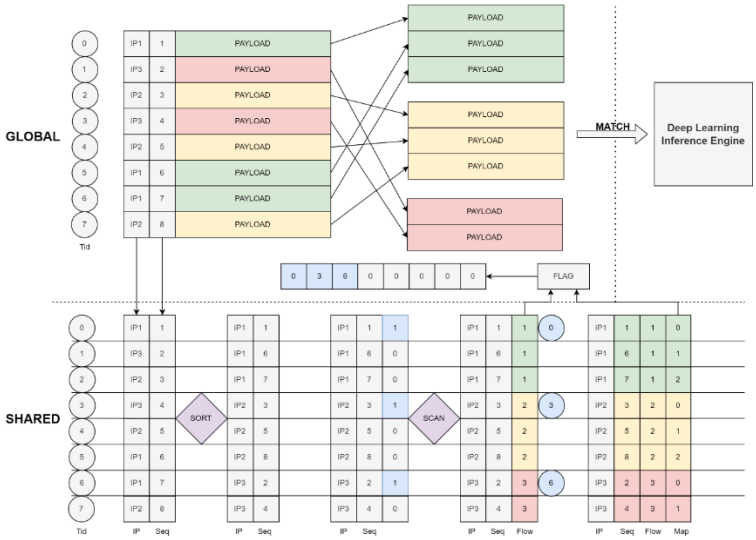

图 基于 GPU 的特征提取（着重展示基于原始数据负载）

当待处理的数据在 GPUFE 上准备完成后，我们映射 global 内存中数据包的 IP 头部和位置 ID（通过线程号+1 得到）复制到 shared 内存上，这避免了后期数据包负载的频繁拷贝。

之后在共享内存中我们对 IP 头部进行处理。如图所示我们计算哈希与偏移计算：每个数据包对应的处理线程计算出基于流 ID 的哈希值，指向 GPU 显存中为该数据流特征预留的存储区（ptr）。结合 MAP 记录的序号，进一步计算出在存储区内的偏移量（offset），确立数据包特征的具体存储位置。特征向量存储：完成上述计算后，特征向量按计算得到的地址（ptr + offset）存储，确保每个数据流的特征向量都能在 GPU 显存中有序地并行生成。特征向量并行生成后，下一步是将这些数据送入推理引擎。为此，我们设计了一个高效的数据流转移策略，确保数据流能够无缝地从特征提取转向推理阶段。特征批次传递：一旦一组数据流的

所有数据包特征提取完成，其存储地址立即被传回 CPU。CPU 随后负责管理这些特征数据，将其送入推理引擎。

选择 TensorRT 推理引擎：鉴于其针对 GPU 优化的高性能特性，我们选用 TensorRT 作为推理引擎。TensorRT 支持广泛的模型和框架，适应高吞吐量需求，有效减少了数据传输开销，保障了推理过程的高效性和低延迟。

通过上述并行化特征映射与推理准备过程，ETCpipe 系统能够有效地处理高速传入的网络流量数据，将其转换为深度学习模型可用的输入特征，并为快速准确的推理决策做好准备。

7.Evaluation

本研究的验证分为两个主要部分，旨在评估 FlowManager 和 GPUFE 两个核心组件的性能。我们通过实验验证 ETCpipe 系统部署 ETC 任务时对网络通信的影响，以及 GPUFE 组件的处理效率。

实验设置：实验环境基于 Dell 服务器，配置包括 NVIDIA V100 GPU 和 BlueField-3 DPU，操作系统为 Ubuntu。我们的实验代码已开源于 GitHub: https://github.com/CHRIS123540/GPU。

端到端测试：端到端的性能测试采用 Iperf 工具，旨在识别系统处理瓶颈，并评估网络设备队列数据包积压对网络协议的影响。相较于先前工作中使用 Pktgen 的方法，Iperf 能够提供更为直接的网络带宽测试结果，从而准确评估 FlowManager 组件是否对网络通信产生影响。

GPUFE 组件测试：在 GPUFE 的性能测试中，我们选用 Pktgen-dpdk 作为发包工具，以消除网络协议可能引入的性能干扰，如乱序或重传，从而更专注于评估特征提取的效率。此部分测试关注于基于原始数据包负载的特征提取——一个之前研究中未充分探讨的领域。我们还将复现基于 CPU 的特征提取方案，并与 GPUFE 组件的性能进行对比，展示后者在处理能力上的优势。

7.2 端到端测试

端到端性能测试的结果如图所示，我们关注的是在应用特定过滤规则（选定流的前 N 个数据包，其中 N=10）下，FlowManager 与 DPU 上原有的 Open vSwitch（OVS）的性能对比。测试旨在评估 ETCpipe 系统对现有网络通信的潜在影响。通过在 Iperf 测试中模拟 2 至 128 个并发流，结果显示 FlowManager 利用 ARM 多核处理能力及硬件加速特性，实现了比原始 OVS 更高的吞吐率。当并发流量超过 128 时，系统对网络通信的影响变得不显著，表明 ETCpipe 系统在高负载情况下能够维持网络通信的效率。

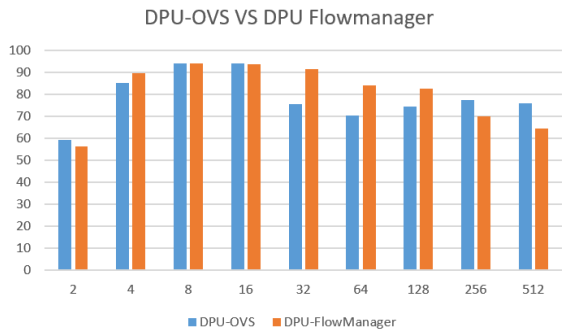

图 e2e 测试

7.3 GPUFE 的处理性能

如图所展示的 GPUFE 与 CPU 处理性能的对比测试结果，我们选取的特征是[Height,Len]，其中 Height 是数据包数量取 10，Len 是需要提取的数据包长度取 1000，这一设置基于当前研究中常见的数据包负载场景。通过 Pktgen-dpdk 工具，我们生成了从 10 条到 100 条流的特征提取任务，并测量了使用 CPU 与 GPUFE 两种方式的执行时间。结果

表明，GPUFE 的处理能力至少相当于 3 个 CPU 核心。性能的提升归因于 GPU 的多核并行加速能力和内存拷贝次数的减少，这一发现突出了 ETCpipe 在特征提取任务上的高效率。

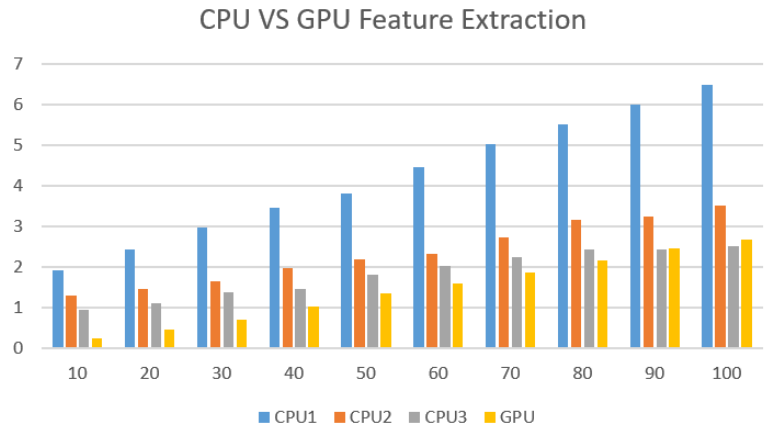

图 GPUFE VS CPU

第七章 Future Work

在本研究中，我们展示了 DPU 的潜在能力，特别是在实现高性能网络通信方面。现有的 FlowManager 通过结合 ARM 处理能力和 RSS 硬件卸载技术，有效地支持了系统的网络处理需求。然而，NVIDIA BlueField-3 DPU 提供了专门的硬件接口（称为 DOCA 硬件语言），它支持基于 eSwitch 的 Match-Action 硬件处理管道，这一方案相比于使用 ARM 处理相同任务能够带来更高的效率。已有的研究表明，通过利用 DOCA 管道，可以在 Linux TC（Traffic Control）上实现 100Gbps 的流量管理。未来，我们计划深入探究 DOCA 技术，以期提升 FlowManager 的处理能力和效率。

此外，ETCpipe 系统目前采用 TensorRT 引擎作为推理后端，专门针对 NVIDIA GPU 进行优化。虽然 TensorRT 提供了出色的推理性能，但考虑到系统设计的多样性和扩展性，我们预计将探索接入其他推理引擎。虽然这可能不会为特定的硬件配置带来最佳性能，但这一步骤对于使系统能够适应更广泛的应用场景和推理平台具有重要意义。通过增加对多种推理引擎的支持，ETCpipe 将能够更加灵活地应用于不同的环境中，为系统的广泛部署和应用奠定基础。。

第八章 Conclude

本研究成功开发了 ETCpipe，一个针对加密流量分类（ETC）任务设计的创新系统，通过利用领域特定加速器深度解耦流量管理与特征提取任务，显著提高了处理效率。ETCpipe 采用 SmartNIC（FlowManager）和 GPU（GPUFE）分别处理这些任务，有效释放了主机 CPU 资源，简化了系统配置，同时实现了加速数据处理。ETCpipe 的设计优化了流量过滤、特征提取和推理执行的每个步骤。FlowManager 利用 SmartNIC 技术实现快速流量管理，而 GPUFE 则利用 GPU 并行处理能力高效提取特征。此外，通过 GPUdirect 技术和高效算法减少了 CPU 和 GPU 间的数据拷贝，进一步优化性能。

测试结果显示，ETCpipe 在保持网络吞吐量的同时，能提供比现有 CPU 解决方案高达三至四倍的特征提取速度，证明了其在效率和性能上的显著提升。ETCpipe 不仅技术上取得进步，还在原始数据包负载分析等实际应用中展示出潜力，为网络监控和管理提供更精准的流量分类和安全分析能力。

总而言之，ETCpipe 为实时流量分析领域带来了创新，填补了现有技术的空白，为未来研究和开发奠定了基础，预示着对该领域的深远影响。

Ref

DPU 链接
https://resources.nvidia.com/en-us-accelerated-networking-resource-library/datasheet-nvidia-bluefield