**ETCpipe: Efficient Encrypted Traffic Classification System Using DPU-Based Networking and GPU-Enhanced Preprocessing**

**Lingxiang Hu**

**Abstract**

随着网络流量模式的日益复杂化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长。现有深度学习网络模型虽在加密流量分类（ETC）中表现卓越，但其实现需消耗大量服务器资源。目前的解决方案，例如FENXI系统，尽管为ETC的实时部署提供了支持，但配置复杂并且侧重于特定模型，忽视了对ETC领域多数模型的支持，从而限制了其通用性。

为应对上述挑战，本研究提出了“ETCpipe”，一种专为ETC任务设计的执行系统。ETCpipe通过将流量管理和特征提取任务解耦至领域特定加速器，有效消除了配置复杂性，提升了处理性能，并释放了主机CPU资源。系统中的FlowManager组件将流量管理任务卸载至DPU，而特征提取任务则由GPU上的GPUFE组件承担。ETCpipe系统利用FlowManager作为网络流入点，将待分析数据交由GPUFE进行特征提取，最终通过推理引擎完成数据分析。我们还解决了GPU处理过滤后数据包的特征提取的难题，通过GPUdirect技术和设计并行算法，将数据直接传输至显存中，加速了特征提取过程。

在NVIDIA V100 GPU和BlueField-3 DPU平台上，我们实现了ETCpipe系统的原型，并验证了其高效性。系统性能评估从网络吞吐量和特征提取效率两个维度进行：与DPU自带的OVS相比，在多种任务上ETCpipe维持了高达100Gbps的网络吞吐率；与FENXI系统相比，在多种任务上ETCpipe的处理效率至少相当于三个CPU核心，并在FENXI的用例上远超3个CPU核心。

本研究的主要贡献在于为不同ETC任务提供了一个深度解耦的系统设计，并通过对原始数据包进行深入的特征提取与测试，不仅简化了系统配置，还显著提升了处理速度和系统的通用性，为ETC任务的实时部署提供了新的解决方案。

**Introduction**

在现代网络架构中，实时流量分析成为了关键组成部分，尤其是随着网络流量模式的复杂化和加密通信技术的广泛应用，对基于深度学习（DL）的流量分析方法的需求持续增加。特别是在加密流量分类（ETC）领域，新兴的深度学习网络模型已证明其卓越性能，成为处理关键网络流量分析任务的强大工具。然而，这些任务在传统服务器上执行时对资源的消耗极大，涉及流量管理、特征提取（亦称数据预处理）和模型推理等多个环节。随着网络速度的显著提升——超过100 Gbps——实时推理系统不仅需保持网络通信的流畅，更必须能够及时处理和分析庞大的流量数据。因此，如何在高速网络环境下有效部署ETC系统，成为了一个亟待解决的问题。

为应对实时分析的需求，业界已开发出多种解决策略。一种常见的方法是将处理任务卸载至网络设备，如n3ic项目通过在SmartNIC上部署模型直接在数据平面执行流量分析，从而实现在网计算。这种方法的优势在于显著减少跨设备通信并释放CPU资源，但由于SmartNIC的资源限制，可部署的模型受到较大约束。另一类解决方案则侧重于使用专用推理硬件，如FENXI系统，它代表了当前技术的最前沿（State-of-the-Art, SOTA）。FENXI通过解耦网络数据平面和流量分析任务，并构建高效的处理流水线，显著提高了系统性能。此方案利用多核CPU执行流量管理和特征提取，同时使用GPU或TPU进行深度学习模型的推理。尽管FENXI能支持较为复杂的模型，其配置的复杂性却成为了一个显著的问题：流水线的资源配置需要手动调整，且要求系统管理员对硬件性能及模型需求有深入的了解以实现最佳系统性能。此外，尽管FENXI进行了基于数据包大小的测试，它在处理原始数据包负载这种更为重型的任务时的通用性尚未得到验证。

在本研究中，我们开发了“ETCpipe”，一个专门为处理异构加密流量分类（ETC）任务设计的执行系统。该系统的核心创新在于其将流量管理和特征提取任务进一步解耦到领域特定加速器上，这一策略显著释放了主机CPU的资源，同时解决了系统配置的复杂性，极大提高了处理速度。更具体地说，我们将流量管理任务卸载到了SmartNIC上的FlowManager。FlowManager负责筛选和过滤数据包，将需要进一步分析的数据包传递给GPU。在GPU上，名为GPUFE的特征提取组件负责提取流量数据的特征信息，最终通过推理引擎（例如针对NVIDIA显卡优化的TensorRT）完成深度学习分析任务。

这种解耦虽然提供了显著的优势，但也带来了一些技术挑战。尽管GPU的高度并行处理能力使其在收集到足够数量的数据包后能有效进行特征提取和推理，它在处理过滤后的数据包时却面临困难，因为GPU本身不擅长直接执行流重组这一任务。这一局限性要求CPU介入以完成数据流的重组工作，随后再将数据拷贝至显存中，这不仅未能完全卸载流量管理任务，还增加了额外的内存拷贝过程，影响了系统效率。

为了克服这些技术挑战，我们优化了数据传输方式并设计了专用的并行处理算法。首先，我们采用了GPUdirect技术，直接将过滤后的数据包从网卡传输至显存中，绕过CPU主存，以减少数据处理的延迟。同时，我们应用了排序和前缀扫描等并行算法完成数据流的重组，实现了高效的特征提取。在NVIDIA V100 GPU和BlueField-3 DPU平台上，我们开发了ETCpipe系统的原型，以网络吞吐量和特征提取效率为关键性能指标进行评估。

通过运用iperf工具进行端到端测试，我们验证了ETCpipe系统在执行多种任务时对网络吞吐量无负面影响，这得益于其高度并行的系统设计。进一步的，我们利用pktgen-dpdk生成的数据包对系统的特征提取效率进行了测评。测试结果显示，在执行多种ETC任务时，ETCpipe的特征提取效率至少达到了FENXI CPU解决方案的三至四倍。针对FENXI的研究用例，ETCpipe的效率远超过三个CPU核心的性能，从而显著提升了整体处理能力。

本研究的主要贡献可以概括为以下两点：

（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。

总之，我们提出了ETCpipe，一个加密流量分类系统解决方案，旨在应对高速网络环境中实时数据处理的挑战。ETCpipe通过领域特定加速器对流量管理和特征提取任务进行了深度解耦，有效释放了CPU资源，同时简化了系统的配置流程。此外，ETCpipe采用了先进的技术如GPUdirect来优化数据传输，提高了数据处理的速度和效率。通过在V100 GPU和BlueField-3 DPU平台上的实验验证，我们证明了ETCpipe能够在不影响网络吞吐量的情况下，显著提高特征提取的效率。这样的系统设计不仅提升了性能，还增强了对多种ETC任务的支持能力，为部署各类ETC深度学习模型提供了一个通用且灵活的解决方案。

**Background and Motivation**

2.1 深度学习在ETC的应用

深度学习加密流量分类（ETC）领域根据输入特征不同，可以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，直接以数据包的原始有效载荷作为输入。另外，由于大语言模型的蓬勃发展，大预言模型也被应用在了ETC任务中，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 DPU 和 GPU

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，其中集成了如ARM处理核心的专用加速器。DPU利用GPUdirect技术加速内存拷贝，PUdirect技术允许数据从网络接口卡（NIC）直接传输到GPU内存，绕过CPU，从而提高数据处理速度。此外，DPU配备了接收端缩放（RSS）卸载功能，R接收端缩放（RSS）是一种分散网络流量的技术，它通过在网卡上计算数据包的五元组哈希来分配流量至多个CPU核心，从而提高处理效率。结构如图X所示

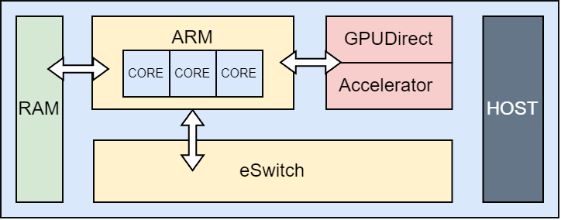


图 Bluefield-3 DPU Architectures

GPU的stream功能支持在同一硬件上并行执行多个任务，这使得不同的计算和数据传输操作能够同时进行，避免资源独占。通过这种方式，特征提取和模型推理等不同过程可以有效地并行部署在单个GPU上，显著提高了加密流量分类（ETC）任务的处理效率和响应速度。

2.3 Motivation

在我们的研究领域中，FENXI系统代表了实时加密流量分类（ETC）处理的一项重要进展。该系统通过CPU来执行流量管理和特征提取任务，并利用TPU/GPU加速模型推理，有效地解耦了数据包转发与模型分析任务。如下图所示，FENXI通常采用1:1:1的配置比例，在特定的研究案例中表现出色。

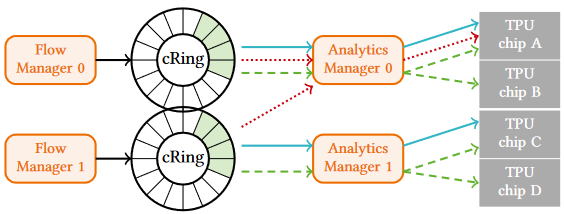


图 FENXI 配置流水线

然而，我们对多种深度学习ETC模型的本地推理实验揭示了系统的显著局限。不同模型因其结构和参数规模差异而导致运行时间的显著不同；此外，在PyTorch框架下，CPU与GPU的推理速度差异也对模型的执行时间产生了显著影响。这要求系统管理员不仅要深入了解各模型的特点，还需熟悉推理硬件和软件环境，以免不恰当的配置成为性能瓶颈。例如，对于推理时间较短的Darknet模型，可能需要更多的CPU核心来处理特征提取；而对于推理时间较长的TSCRNN模型，配置过多的CPU核心则可能造成资源浪费。如图所示

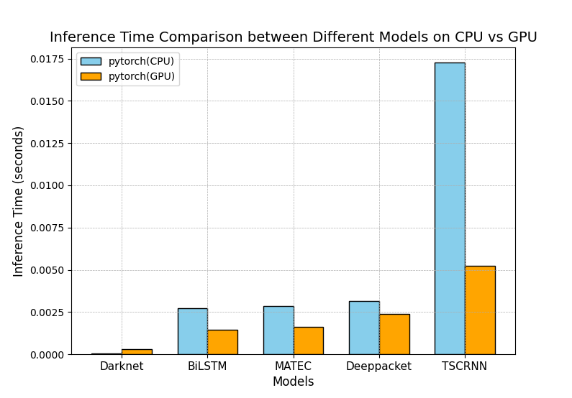


图 不同模型在不同平台的执行推理

此外，FENXI的研究主要限于分析前K个数据包的长度，没有考虑基于原始负载数据的更多样化分析。这种研究的局限性削弱了模型处理原始负载数据时的适用性，并限制了系统的广泛应用。原始负载数据是ETC任务中的关键输入特征，其深入分析对理解加密流量的本质至关重要。

因此，FENXI系统配置的复杂性和在通用性上的不足，是我们研究的动机。

**Systems Design**

3.1 设计目标

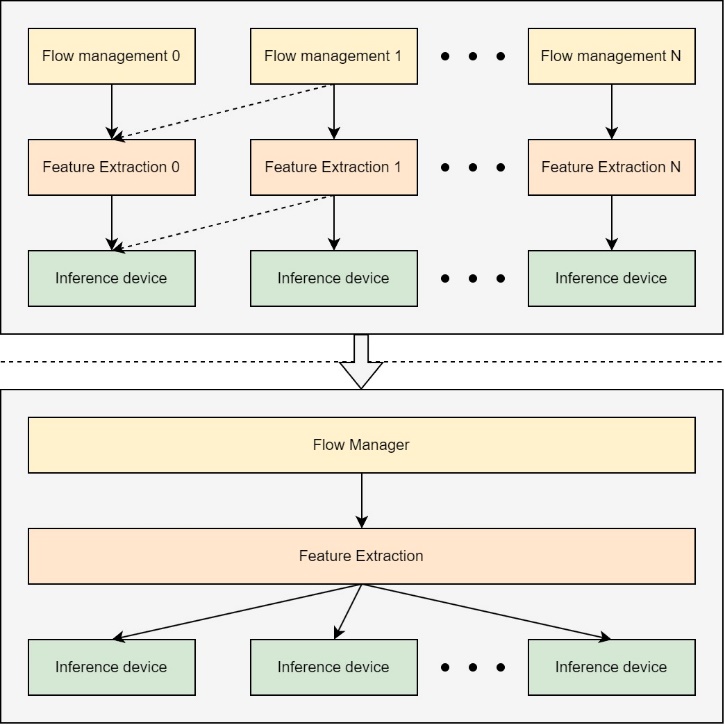


图 主要观点

本研究旨在彻底解耦流量管理与特征提取任务，并最终通过集成硬件加速器并利用推理框架，实现一个由深度学习驱动的加密流量分类（ETC）系统。此解耦策略的提出基于对现有ETC系统配置复杂性的关键分析，特别是在系统需要适应多样化深度学习模型的环境中。

从动机中我们可以得出，传统ETC系统因配置的难题而难以灵活应对多变的模型需求。例如，当推理模型或特征提取算法需要更新时，整个系统的配置过程需要重新进行，这不仅耗时且易于出错，还大幅增加了系统的维护负担。为了解决这一问题，本研究提出将流量管理任务专门交由DPU处理，并将特征提取及模型推理任务专门交由GPU执行。这种硬件层面的解耦不仅简化了系统配置，提高了处理效率，也增加了系统对不同模型调整的灵活性。

此外，该解耦方法还为系统的扩展性和可升级性提供了便利。各硬件组件可以独立进行优化和升级，无需担心对整个系统造成干扰。因此，这种策略直接解决了现有系统中存在的通用性和配置复杂性问题。

首先，我们设计需要实现快速特征提取，这需要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。在数据传输方面，改进传输的方式，降低延迟和带宽占用，实现更高效的特征构建。

同时，为了提升系统对各类ETC应用的适用性，我们将扩展特征分析至基于原始数据包负载，填补现有系统在原始数据处理方面的空白，并确保即使在系统升级活ETC任务改变后依然无需关注配置问题，减少管理员的干预需求。

在数据中心部署系统时，我们的设计需要确保系统能够处理高吞吐量的数据流，从而维护网络服务质量（QoS）和优化用户体验。为此，我们专门在数据处理单元（DPU）上开发高效的算法和组件，这些设计将网络数据管理与推理任务有效分离。通过这种方式，我们确保系统的数据处理和模型推理操作不会相互干扰，从而保持不变的高网络传输速率和响应能力。

3.2 系统架构概述

A diagram of a computer

Description automatically generated

图 ETCpipe 系统架构

本节介绍ETCpipe系统的架构，该架构专门设计来解耦流量管理和特征提取任务，提高处理效率保证网络正常。系统主要由三个部分构成：流量管理器（FlowManager）、特征提取器（GPUFE）、以及推理引擎。FlowManager负责在数据处理单元（DPU）上筛选和管理数据流；GPUFE在图形处理单元（GPU）上执行特征提取；推理引擎用于进行分析得到结果。

首先，网络数据包经过流量管理器，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。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将给特征提取器进行转发，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。

FLowManager转发的符合条件的数据包将进入GPUFE，一个部署在GPU上的特征提取器。该组件利用GPUDirect技术，直接从流管理器将数据包高效传输至GPU显存，从而降低延迟并提高数据处理速度。GPUFE并行执行特征提取任务，并利用专门的算法从数据包中提取所需特征。更多关于特征提取算法的细节将在后续章节进行详细讨论。

提取的特征随后被送入推理引擎进行分析，以生成最终结果。在本系统中，推理引擎可部署于CPU、GPU或TPU上，但考虑到我们的原型是基于Nvidia V100 GPU，我们选择使用TensorRT推理框架，它优化了基于Nvidia GPU的推理性能。

综上所述，ETCpipe系统处理数据包的流程概述如下：首先，数据包首先经过至FlowManager，FlowManage判断数据包是否符合预设的条件；那些不符合条件的数据包会被直接转发，而符合条件的数据包则选定进行进一步的分析。这些数据包随后被传递至GPUFE，之后GPUFE进行提取特征。提取的特征随后输送至推理引擎，负责进一步的数据分析与结果生成。这种系统的解耦设计，极大地简化了管理层面的操作复杂性，允许管理人员专注于数据分析，增强了系统的灵活性和效率。

**4. Implementation**

**4.1 FlowManager**

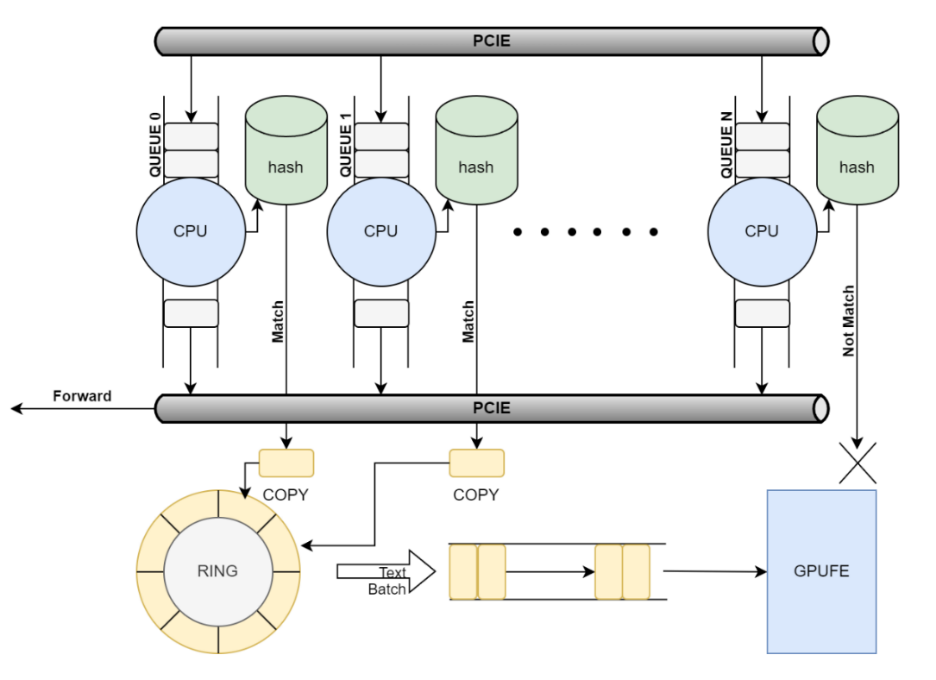


图 FlowManager 框架

FlowManager是以数据平面开发套件 (DPDK) 实现的。FlowManager组件作为ETCpipe系统处理的初始节点，负责高效管理网络通信，并确保特征提取任务与网络通信的有效隔离，防止二者相互干扰。本组件的核心功能是实时监控和管理流量状态，以满足多样化的ETC任务需求。FlowManager对入站流量进行初步筛选和必要过滤，从而为GPUFE组件的后续特征提取工作提供数据准备。如图X所示，流量处理的初始步骤包括将数据包均匀分配到多个处理队列，每个队列根据与当前部署模型相匹配的特定过滤规则进行数据筛选。那些符合筛选条件的数据包随后被复制至DPU的环形队列，并批量传输至GPUFE。

具体来说，FlowManager针对需要分析流中前N个非ACK-FIN数据包的ETC任务，有效地结合DPU的软件和硬件资源执行筛选算法。该系统配置有多个处理队列，每个队列由单个ARM核心控制，确保了数据处理的高效性。此外，利用接收端缩放（RSS）功能在硬件层面计算数据包的五元组哈希值，显著减轻了每个ARM核心的计算负荷，并确保了数据包在各网卡队列中的均匀分布。

当数据包到达系统时，它们首先通过RSS功能被自动分配到指定的处理队列。在这些队列中，每个数据包根据其五元组哈希值进行流状态匹配查询，该查询利用哈希桶来跟踪和确定数据包在特定流中的位置。如果数据包被识别为目标任务所需的前N个非ACK-FIN数据包之一，该数据包会被复制并存入DPU上预配置的环形队列中，并同时更新哈希桶中的流状态信息，以备后续处理，而原数据包则继续沿常规路径进行转发。不满足条件的数据包同样按照常规路径直接转发，不涉及哈希桶的状态更新。

这些环形队列定期从各处理队列收集筛选后的数据包，并将它们批量发送至GPUFE进行进一步的特征提取处理。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。

**4.2.** **GPUFE**

在ETCpipe系统中，特征提取环节对于将网络流量数据转换成满足深度学习模型处理的格式至关重要。此环节接续FlowManager的输出，GPUFE模块负责对筛选后的数据进行预处理。通过高效的数据处理算法进行特征提取，GPUFE将原始流量数据转化为结构化的特征向量，为深度学习模型提供准备好的输入数据。

6.1 技术挑战及解决策略

在ETCpipe系统中，利用GPU进行特征提取引入了复杂的技术挑战，特别是关于并行特征提取的实现及GPU计算资源的有效管理。并行特征提取的主要挑战在于流的同步识别——确定并处理属于同一流的多个数据包。在传统的CPU计算模型中，这一任务通过串行处理每个数据包并连续更新流状态来实现。然而，在GPU的并行计算环境中，直接采用类似的串行方法会导致效率低下。具体来说为维持数据一致性，需要在GPU上实现锁机制，保证同一时刻只有一个线程更新流状态。这种处理方式不仅降低了GPU并行计算的效率，还增加了因锁竞争而导致的资源浪费。此外，数据包从FlowManager到GPU的传输也涉及显著的开销。数据首先需要从网络接口卡移至主机的主存中，接着通过CPU指令进行内存拷贝至GPU。这种传输模式增加了额外的延迟和处理负担，有时直接在CPU上完成特征提取反而更加高效。

GPU资源的竞争也构成了一个显著的挑战。特征提取虽为系统的关键组件，但与此同时如果推理引擎也需访问相同的GPU资源，可能引发资源竞争。所以尽管增加持久化kernel及多线程块可以提升特征提取的效率，但过多的资源占用可能会限制推理引擎的性能，影响整个系统的运行效率。因此，必须在确保性能的同时合理规划GPU资源的分配，以避免过度占用和资源冲突。

为有效应对上述挑战，我们采取了一系列创新措施，主要从数据传输模式和并行算法两个关键方面入手。首先，通过GPU Direct技术的应用，我们实现了数据包从网络接口卡（NIC）直接传输到GPU内存的过程，显著降低了CPU到GPU的数据拷贝需求，从而减少了数据传输的开销。这一技术的使用为高效并行算法的实施提供了坚实的基础，使GPU能够直接处理输入数据，避免了传统的内存拷贝延时。在GPU上，为了充分发挥其并行处理能力而不引入常规的加锁机制，我们开发了一种创新的并行算法专用于特征提取。该算法核心在于实现了高效的流重组，通过这种方式，我们可以无需锁的干预即可确定每个数据流的状态。这不仅提高了计算效率，也避免了因并发访问同一内存区域而导致的资源冲突和延迟。

此外，为了解决GPU资源争抢的问题，我们实施了资源分配策略，明确划分了不同任务所使用的GPU资源。在系统设计中，GPUFE模块被限制只能启动包含1024个线程的线程块来处理数据包，确保每个任务都能获得必要的资源而不影响其他并行任务。这种策略既最大化了GPU的处理能力，同时也避免了资源的过度使用和竞争。

这些技术解决方案的详细实现和效果将在第4.2节进行深入讨论。

4.2GPUFE组件实现

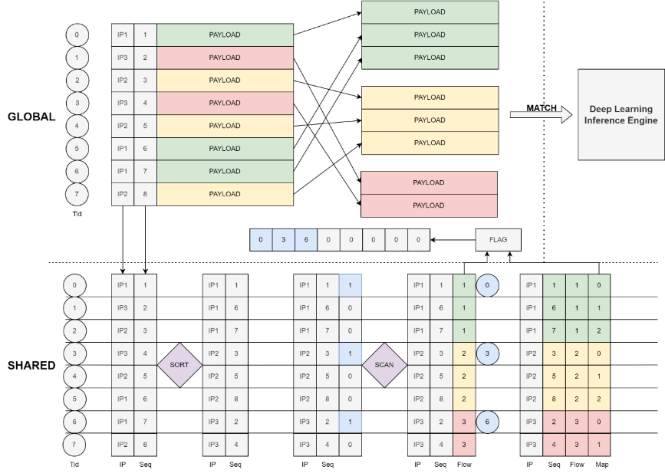


图 基于GPU的特征提取

在ETCpipe系统中，GPUFE模块通过GPU Direct技术直接从网络接口卡（NIC）接收数据，该技术允许数据包绕过CPU直接进入GPU的显存，从而减少了传统数据传输过程中的延迟和带宽占用。此举是为了充分利用GPU的高并行性来处理数据包的特征提取任务。

在ETCpipe系统中，GPUFE模块对并行特征提取的实施分为两个关键步骤：首先是在共享内存中实现流重组以确定数据流的状态；其次是在全局内存中根据流重组的结果进行特征的结构化处理。为了展示我们的算法流程，尽管实际操作可能涉及高达1000个数据包，我们在此以8个数据包作为示例，如图所示。数据包经由FlowManager筛选后，通过GPU Direct技术传输至GPU的全局内存。

为提高计算效率，流重组过程被迁移到效率更高的共享内存中执行。在共享内存中，我们初始化以下存储结构：(1)IP: 数据包的五元组信息。(2) Seq: 数据包的到达顺序。(3)Flow: 标识数据包所属的流。(4)Map: 记录数据包在其所属流中的序号。(5)Flag: 用于标记每条流的起始地址。(6)Temp: 用于临时存储计算结果。

处理流程从全局内存中提取数据包的IP信息和到达顺序到共享内存的IP和Seq列。接着，我们采用基于GPU结构优化的双调排序算法对IP列进行排序，Seq列随IP排序而相应调整，以确保流内数据包的顺序正确。

接下来，通过比较相邻的IP值，若当前IP与前一IP不同，将此位置在Flag数组中标记为新流的开始，并在Temp列对应位置标记为1；对于相同的则标记为0。对于第一个数据包，由于没有前一个IP进行比较，我们将其直接标记为1。

利用前缀扫描算法处理Temp列的值，计算结果存入Flow列，从而将每个数据包归属到特定的流。为确定数据包在其流中的具体位置，通过将Flag数组的起始地址值与线程标识号(tid)相减，得到Map值。

流重组完成后，对于任一数据包，通过比对其Seq号、Flow和Map值，可以准确判定该数据包在特定流中的序号。最后，根据需要提取的特征，如payload或数据包长度，可以并行地进行特征提取和结构化处理。整个的算法如table所示。

针对资源分配的问题，我们通过在性能和资源利用上进行折中以实现资源的合理分配，仅分配1024线程块给GPUFE，并且绑定在GPU的某个stream上以实现多任务并行。

在数据流的所有数据包经过特征提取之后，接下来的步骤是进行推理分析。虽然理论上推理引擎可以部署在任何硬件平台或使用任何软件框架，但在本系统中，我们选择了TensorRT作为推理后端。这一选择主要基于TensorRT对NVIDIA GPU的高度优化，使其在我们的应用场景中表现出色。具体操作中，已提取并结构化的特征数据直接通过GPU地址传递给TensorRT，完成最终的推理和分析。

**5.Evaluation**

为了全面验证ETCpipe系统在实际生产环境中的性能，我们从两个关键维度进行评估：一是系统在100 Gbps网络环境下的表现；二是在特征提取任务上的处理速率。

5.1实验设置

实验环境：本实验使用了NVIDIA V100 GPU和BlueField-3 DPU作为硬件支持，操作系统选择了Ubuntu 22.04。所有实验所用代码已开源，并可在GitHub上访问：<https://github.com/CHRIS123540/GPU>。

实验方法：

网络性能测试：我们使用iperf工具来模拟100 Gbps的网络环境，并评估ETCpipe系统在此环境下的网络性能。

特征提取性能测试：在特征提取方面，我们采用Pktgen-dpdk来生成网络流量，以测试系统处理此类数据的速率和效率。

5.2 FENXI

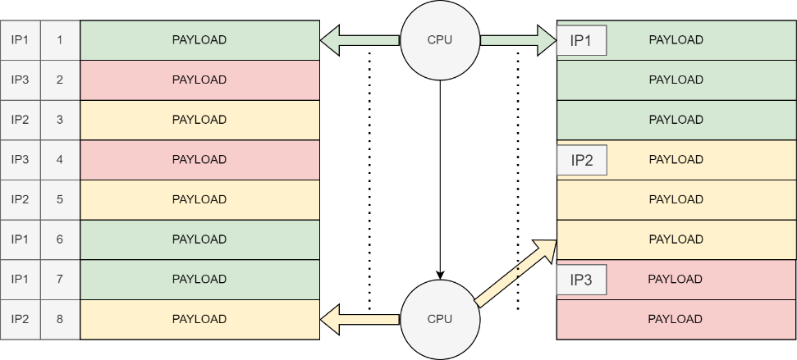


图 FENXI

在之前的工作中，特征提取通常通过CPU完成。尽管这一方法在早期的ETC系统中，例如“FENXI”系统，得到了广泛应用，但它在处理效率、资源利用和系统配置方面存在明显的局限性。一个典型的基于CPU的特征提取过程可以通过如图所示的方式进行描述，以8个数据包为例。在此过程中，CPU需逐一处理接收到的穿插到达的不同数据流中的数据包。CPU从第一个数据包开始，串行地识别各个数据流并逐一提取特征。一旦某个数据流的特征提取完成，其特征便被送往后续的推理模块。

如第二章所述，尽管部署多个CPU核心能在一定程度上提升处理性能，设计一个高效的流水线配置仍然是一个技术挑战。这要求系统管理员不仅要深入了解深度学习模型的需求和硬件特性，还需要精细地优化资源配置以实现最大化的利用率。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极大地限制了处理速度，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，进一步增加了系统配置的复杂性。

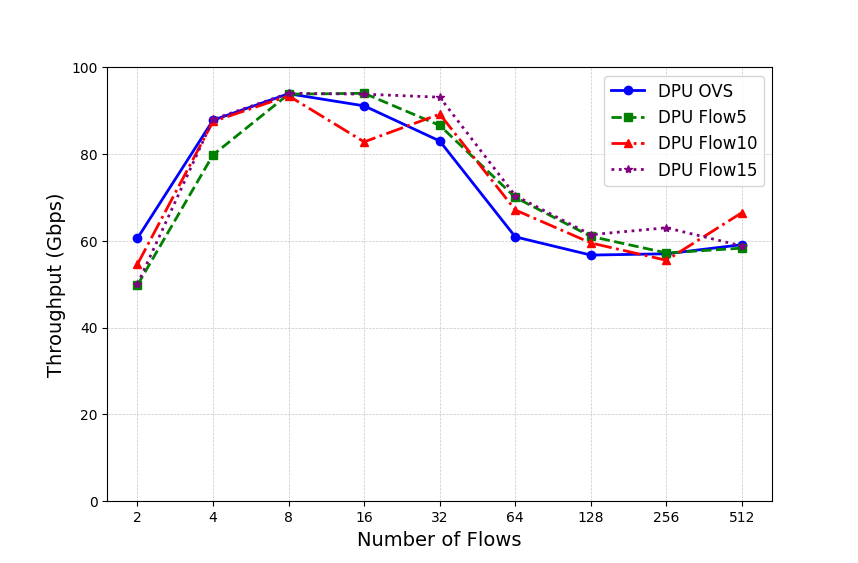
我们对这样的传统模式进行了复现，并将其作为基准与我们的特征提取器进行比较，以展示我们系统的高效性。

5.3 100gbps的网络

首先，确保我们的系统在实际生产环境中部署时不会影响现有网络通信能力是至关重要的。以往的研究在测试网络吞吐量时常采用重放PCAP文件的方法，虽然这可以实现100 Gbps的发包速率，但它并不能充分反映真实网络环境的动态。特别是在考虑到TCP协议中的拥塞控制机制时，单纯的发包测试并不足以揭示网络协议或网卡队列拥塞的真实情况。为了识别系统的处理瓶颈并评估队列积压对网络协议的影响，我们采用了端到端性能测试工具Iperf，它能提供直接的网络带宽测试结果，与Pktgen相比，Iperf能更准确地评估我们的FlowManager组件是否对网络通信产生了影响。我们选择与DPU自带的DPU-OVS进行吞吐量比较，而不是与FENXI系统比较，因为只有与原有的产品级OVS比较，我们才能真实地反映出系统的受影响程度。

在我们的实验中，对ETCpipe系统的网络性能进行了定量评估，确保所有测试在2至512个TCP并发流的范围内进行，并统一进行了30秒的测试时长，以消除短期波动对数据的影响。实验结果，如所附图示，显示了DPU-OVS在不同网络负载下的吞吐量行为。此行为呈现出初期随流数增加而上升的趋势，超过某一点后随流数增加而下降，最后在较高流数时平稳下来，形成了实验的基准线。

此外，实验中还包括了使用ETCpipe系统，设定不同的特征提取阈值（分别为5、10和15个数据包）的三个实验条件。这些条件下的测试目的是精确测定ETCpipe系统在不同种类数据包过滤工作下的性能表现。在经过Iperf测试模拟的不同流数量下，ETCpipe系统展现出了通过ARM多核处理和硬件加速所带来的吞吐量维持能力。这一发现是显著的，表明即便在多达512个并发流的极端网络负载条件下，ETCpipe系统能够有效地保持网络通信的稳定性和效率。

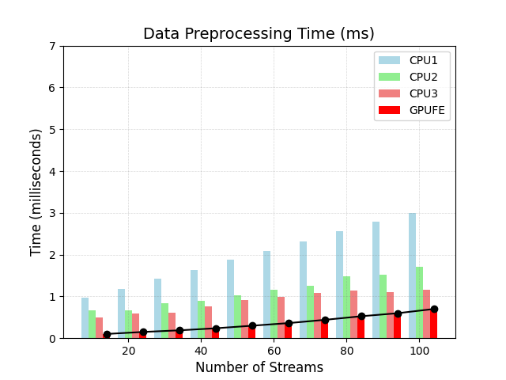


100gbps 网络下的测试

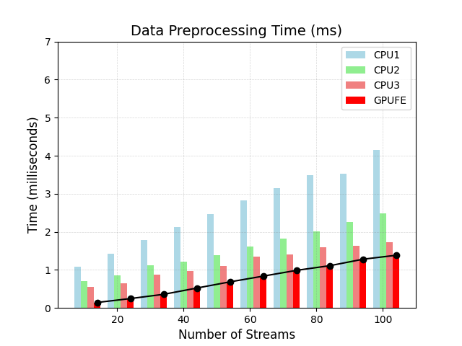
5.4特征提取的速率

特征提取速率的评估是对ETCpipe系统性能测试的另一重要方面。为了进行比较分析，我们复现了FENXI系统中采用的特征提取策略，并针对单核CPU、双核CPU以及三核CPU配置进行了特征提取的性能测试。在这部分的性能测试中，我们采用了Pktgen-dpdk作为发包工具。此选择基于需要排除由于iperf工具可能引入的CPU和GPU之间的处理差异——例如乱序或重传现象——这些现象可能会在特征提取时间计算中引入网络传输延迟的误差。因此，我们专注于评估特征提取的纯粹效率，而非其在网络传输过程中的表现。

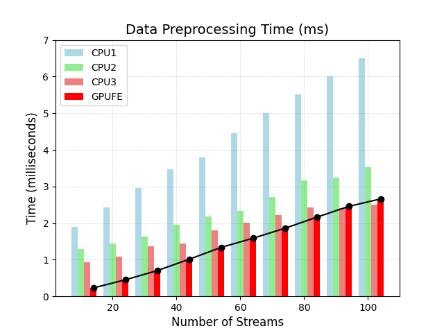
我们的测试初步集中在基于原始数据包负载的特征提取性能上——这是ETC任务中之前研究尚未充分展示的一个维度。对比测试结果展示了GPUFE与不同CPU配置下处理性能的差异，如附图所示。我们选择的特征配置包括[5,500]、[5,1000]和[10,1000]，其中第一个数字表示流所需的数据包数量，第二个数字表示每个数据包所包含的字节数，这一测试配置反映了当前研究中常见的数据包负载场景。利用Pktgen-dpdk生成10到100条流的特征提取任务，我们测量了在使用传统CPU方法和ETCpipe系统中进行特征提取的执行时间。实验结果揭示，在特征规格为[5,1000]和[10,500]时，ETCpipe的特征提取速度超过了使用三个CPU核心的配置，这得益于数据传输优化和并行处理策略的有效实施。而在[10,1000]的特征规格下，三核CPU配置的性能与ETCpipe系统相当。



5 500

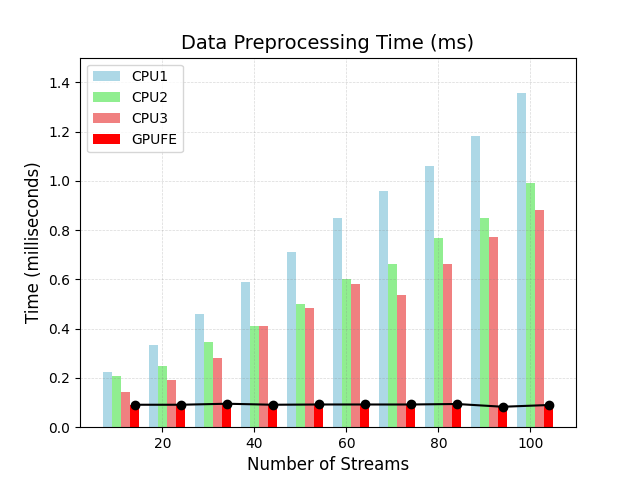


10 500



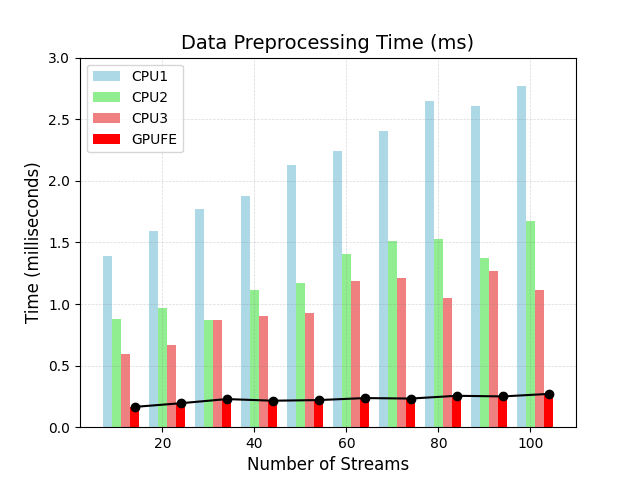
10 1000

我们进一步对FENXI研究中的特定案例进行了再现和性能对比分析。在FENXI系统中，性能评估基于流的前10个数据包大小，采用这一特定的数据结构作为特征提取的输入。经过实验验证，如图所示，ETCpipe在特征提取的速率上显著优于采用三个CPU核心的配置。这一显著的性能提升归因于在ETCpipe处理流程中减少了数据拷贝操作的需求，从而优化了数据处理路径。



FENXI

在ETCpipe系统的性能研究中，我们补充考察了基于数据包的特征输入格式——这不仅是流处理的一种特殊情形，而且也是深度学习在ETC任务中常用的输入特征。具体地，我们采用了数据包的前786个字节作为特征提取的输入。实验结果如图所示，ETCpipe在处理此类特征提取任务时的速率显著高于三核CPU配置的速率。对于这类计算密集型而数据量相对较小的任务，性能的提升主要得益于在数据处理流程中实现的数据拷贝次数的显著降低。



1 786

图 GPUFE VS CPU

**第六章 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。。

**第七章 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