Real-time Deep Learning based Traffic Analytics

Massimo Gallo, Alessandro Finamore, Gwendal Simon, Dario Rossi
Huawei Technologies Co. Ltd - first.last@huawei.com

1 INTRODUCTION

The increased interest towards Deep Learning (DL) technologies has led to the development of a new generation of specialized *hardware accelerator* [4] such as Graphic Processing Unit (GPU) and Tensor Processing Unit (TPU) [1, 2]. The integration of such components in network routers is however not trivial. Indeed, routers typically aim to minimize the overhead of per-packet processing (e.g., Ethernet switching, IP forwarding, telemetry) and design choices (e.g., power, memory consumption) to integrate a new accelerator need to factor in these key requirements. The literature and benchmarks on DL hardware accelerators have overlooked specific router constraints (e.g., strict latency) and focused instead on cloud deployment [3] and image processing. Likewise, there is limited literature regarding DL application on traffic processing at line-rate.

Among all hardware accelerators, we are interested in *edge TPUs* [1, 2]. Since their design focuses on DL *inference*, edge TPUs matches the vision of operators, who consider running *pre-trained* DL models in routers with low power drain. Edge TPUs are expected to limit the amount of computational resources for inference and to yield a higher ratio of operations per watt footprint than GPUs.

This demo aims to investigate the operational points at which edge TPUs become a viable option, using traffic classification as a use case. We sketch the design of a real-time DL traffic classification system, and compare inference speed (i.e., number of classifications per second) of a state-of-the-art Convolutional Neural Network (CNN) model running on different hardware (Central Processing Unit (CPU), GPU, TPU). To constrast their performance, we run stress tests based on synthetic traffic and under different conditions. We collect the results into a dashboard which enables network operators and system designers to both explore the stress test results with regards to their considered operational points, as well as triggering synthetic live tests on top of Ascend 310 TPUs [1].

2 SYSTEM VIEW

We depict in Fig. 1 a high-level view of in-network DL analytics in a general purpose server architecture. Traffic captured by the Network Interface Card (NIC) is handed over to the CPU. For the sake of clarity, we represent a single NIC and CPU, but multiple NICs and Non-Uniform Memory Access (NUMA) nodes could be supported with such architecture.

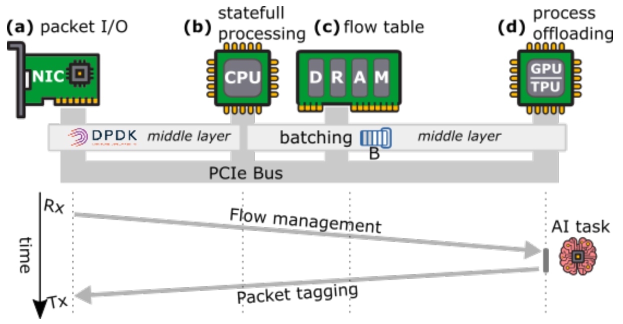

Figure 1: High-level system view for real-time traffic analysis on classic Von Neumann architectures.

Stateful processing (e.g., flow management) needs access to off-chip memory (DRAM) to prepare input to the DL algorithm, which can be performed by general purpose CPU or by dedicated GPU/TPU accelerators. Results of the DL inference may need to alter the packet (e.g., tagging), which can be done by the CPU using Direct Memory Access (DMA) before instructing the card to send the packet.

2.1 Prototype Design

In the prototype, DPDK [6] interacts with the NICs for packet I/O **(a)**, whereas CPU **(b)** and memory **(c)** maintain a flow table with basic per-flow metrics (e.g., number of bytes and packets) and the *features* required for inference. We resort to "early traffic classification" [5] where model input consists of packet size and direction of the first K packets of a flow: hence, in terms of memory each flow occupies an array of size $K+1$, since the result of the classification occupies one additional slot. DL inference task can then be either executed by the CPU or offloaded to either a GPU or a TPU **(d)**.

The implemented DL model is a CNN composed of a stack of convolutional and max-pooling layers, plus a dense layer with softmax activation. The CNN model can identify 200+ applications and has been trained with Tensorflow v1.15 on NVidia V100 GPUs, with real traces from enterprise and residential access networks.

In addition to the flow-table, CPU and memory maintain a list of flows ready to be analyzed. We leverage *batching*[1] to optimize DL operations and communication via PCI-Express. After at least B flows are in the queue, the system triggers model inference for the batch and collects results in the flow table for tagging subsequent packets of the analyzed flows.

[1]A *batch* contains the information of B flows. The design of recent accelerators and DL framework enables fast parallel processing of the B flows.

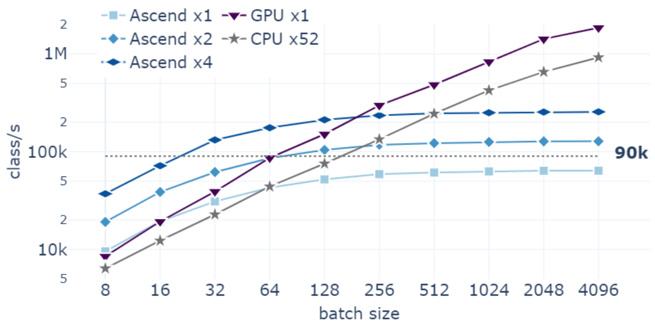

Figure 2: Number of classifications per second for different batching configurations.

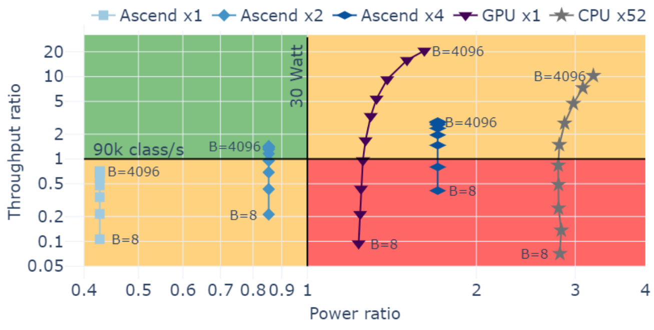

Figure 3: Comparing throughput and power budget across different configurations.

3 DEMONSTRATION

The prototype runs on a server, equipped with two ConnectX-5 single port 100Gbps Mellanox NICs and Intel Xeon Platinum 8164 CPUs @ 2.00GHz (L1/L2/L3 caches 32 data+32 instruction)/1024/36608 kB). As DL offload hardware, we use either an Huawei Atlas 300 TPU (equipped with 4× Ascend 310 chips), or an Nvidia V100 GPU. To provide a fair comparison across the CPU and GPU cases, we do not employ the native Huawei Mind Studio stack and rather cross-compile the original TensorFlow model for the Huawei Atlas engine. In the demonstration we focus on DL processing performance analysis, whereas a deeper analysis of the flow processing part is reserved as future work.

3.1 Performance comparison

The prototype helps in contrasting the performance of DL processing with respect to multiple criteria such as latency, throughput (classifications per second), and power usage. It also highlights the impact of system parameter settings such as batch size B.

Throughput. The demonstration highlights the capacity of DL accelerators to sustain traffic classification at line-rate. The dashboard includes a visual such as Fig. 2 to report the maximum throughput achievable across different batching configurations. During system design phase, operators and manufacturers can derive from this graph the admissible throughput in byte per second by adjusting inner traffic and NIC characteristics (i.e., average flow length, packet size, and link capacity).

Latency. We estimate the average latency between the time at which the NIC receives the K^{th} packet of a flow and the time it gets the flow label. The demonstration reveals that large batch size may be impractical due to the latency introduced by batch completion. While a flow waits for the other $B-1$ flows to arrive, new arriving packets remain untagged, and worse, the flow may be classified *post-mortem*. System designers are thus likely to set small batch sizes, at the price of lower throughput processing.

Power Usage. The demonstration also presents the power consumption of the accelerators depending on the batch size. This performance criteria also matters for system designers who decide router architecture. To illustrate this trade-off, the dashboard includes a visualization like Fig. 3 comparing power (x-axis) and classification rate (y-axis). We normalized performance with respect to reference values (offered as control widget by the dashboard, but not visible in the figure). The shaded areas further highlight configurations offering desirable (top-left corner) and to be avoided (bottom-right corner) operational areas. For instance, only one hardware configuration (i.e., 2 Ascend 310) with a couple of settings can meet the requirement of a challenging configuration (90k class/s and 30W), while the other hardware configurations fail in at least one dimension (i.e., light red corners).

3.2 Demo workflow

The demo is a Voila dashboard[2], using the plotly library and jupyter widgets for interactions. As shown in the short video accompanying this submissions, demo users will be able to observe and interact with system and scenario parameters to better grasp the impact of each considering both live and pre-computed results.

REFERENCES

[1] 2020. Ascend 310 chip. https://e.huawei.com/se/products/cloud-computing-dc/atlas/ascend-310. (2020).
[2] 2020. Google Coral. https://coral.ai/products/. (2020).
[3] D. Crankshaw et al. 2017. Clipper: A Low-Latency Online Prediction Serving System. In *USENIX NSDI*.
[4] A. Reuther et al. 2019. Survey and Benchmarking of Machine Learning Accelerators. In *IEEE High Perf. Extreme Comp. (HPEC)*.
[5] L. Bernaille et al. 2006. Traffic Classification on the Fly. *SIGCOMM Comput. Commun. Rev.* 36, 2 (April 2006), 23–26.
[6] Intel. 2020. Data Plane Development Kit. http://dpdk.org/. (2020).

[2]https://github.com/voila-dashboards/voila