CCPA: A Concurrent Content Processing Architecture for Hardware Firewalls

Shuangliang Chen
UIUC

Saptadeep Pal
Etched

Rakesh Kumar
UIUC

***Abstract*—Hardware firewalls are critical components of today's data centers and large enterprises. However, these firewalls demonstrate poor utilization for network traffic that is dominated by a small number of large sessions (elephant flows), sessions with high network flow bandwidth. For these large sessions (and, in fact, all sessions), we observe that the bottleneck is that the processing of all packets is serialized on a single data processing card (DPC) of the firewall to support per-connection consistency, lowering overall utilization. We make a novel observation that only the stateful inspection phase of packet processing truly needs to be serialized – the content inspection phase of packet processing, which dominates the overall processing time, can be parallelized across multiple DPCs without impacting correctness. Based on this observation, we propose CCPA, a novel architecture of hardware firewalls where the stateful inspection for all packets in a session is first performed sequentially on a dedicated processor before the packets are sent to DPCs for concurrent content inspection. By addressing the utilization bottleneck, CCPA improves the average firewall throughput by 4.29x - 14.3x when using an optical backplane.**

I. Introduction

Next-generation firewalls (NGFWs) [13] serve as the front-line defense in network security today, crucial for protecting network integrity by monitoring and controlling network traffic based on predetermined security rules as well as performing application-level analysis of every packet. Hardware implementations of these firewalls that offer high throughput are, in particular, critical components of today's data centers and large enterprises, especially as the security threats in these domains become more dire [30] [38] [47].

Unfortunately, high-throughput hardware firewalls are often expensive, power-hungry, and bulky (Table I). This is due to the fact that they often consist of multiple data processing cards (DPCs), network processing cards (NPCs), and load balancers ((Figure 1(A)) [34] [19] with each card occupying multiple Rack Units (RUs) and often hosting multiple high end and power-hungry processors, FPGAs, and ASICs, to support at high throughput and low latency the necessary NGFW networking and security features, such as on-the-fly SSL decryption/encryption [42], Carrier-Grade NAT [41], IPsec VPN [9], pattern/signature matching [46], SD-WAN [53], etc. The high monetary, volume, and power overheads mean that high throughput hardware firewalls must demonstrate high utilization and scalability to limit overall costs.

In this paper, we show that the architecture of today's high-throughput hardware firewalls fundamentally limits the utilization that these firewalls can achieve for network traffic that consists of elephant flows – large and long-lasting sessions that consume a substantial portion of network bandwidth – and mice flows – small and short-lived sessions [32] (Section II). As today's hardware firewalls keep track of the state of a network session locally on every data processing card (DPC) (Figure 1(B)), packets from a single high-bandwidth session will be mapped to the same data processing card to ensure that state access and updates are sequentially consistent [35] [17]. This conservative approach simplifies firewall design by ensuring that packets are always processed and delivered in order, a key requirement of the TCP protocol (out-of-order packets are treated as a sign of congestion, and throughput is reduced accordingly [27].) It also obviates the need for complex synchronization across DPCs. However, this conservatism produces a utilization bottleneck – uneven utilization among DPCs and low overall utilization. Since many real-world network traffic traces actually consist of elephant flows (up to several hundred Gbps) and mice flows [21] [31] (as low as a few Kbps), we show that this poor utilization causes a significant gap between actual throughput and peak throughput of the firewalls on these traces, leading to poor overall efficiency and scalability (Section II).

Prior work on packet processing [49] has observed an analogous uneven utilization problem among the parallel processing elements (PEs) of a single DPC. Because of the stateful nature of TCP, a naive implementation load balances the packets of the same session to the same PE, where each PE keeps track of a private session table. When facing low entropy traffic (packets only belong to a small number of sessions, such as file transfer and model training), the PE arrays can become underutilized. To solve this problem, a single, large session table is stored in a centralized memory location on the DPC, and load is balanced among the PEs of the DPC in a stateless manner. During processing, PE can access this centralized session table to complete processing. However, this solution cannot be applied to solve the uneven utilization problem among the DPCs of a firewall. Implementing a centralized session table that each DPC reads and writes to can cause significant performance degradation (Section V), because the backplane bandwidth (800 Gbps / DPC) is not only lower than the on-board bandwidth in the packet processor's case (3200 Gbps / PE), but the packet throughput is also much higher (often $>$10x) for rack-scale firewalls than the individual DPCs. An entirely different solution would be needed.

Table I
CHARACTERISTICS AND COST OF TODAY'S HIGH-END HARDWARE FIREWALL

Model	Vendor	# DPCs	# NPCs	# LBs	Firewall TP (Gbps)	NGFW TP (Gbps)	Cost ($)	Power (W)	Volume (RU)
FG-7121F [16]	Fortinet	10	2	2	1,890	520	1,578,199	9,820	16
FG-7081F [16]	Fortinet	6	2	2	1,890	312	589,500	7,370	12
PA-7500 [37]	Palo Alto Networks	5	2	2	1,500	1,440	-	8,200	14
PA-7080 [36]	Palo Alto Networks	6	4	1	595	313	2,132,372	7,000	19
SRX5800 [24]	Juniper	8	2	1	3,360	699	196,320	8,200	16

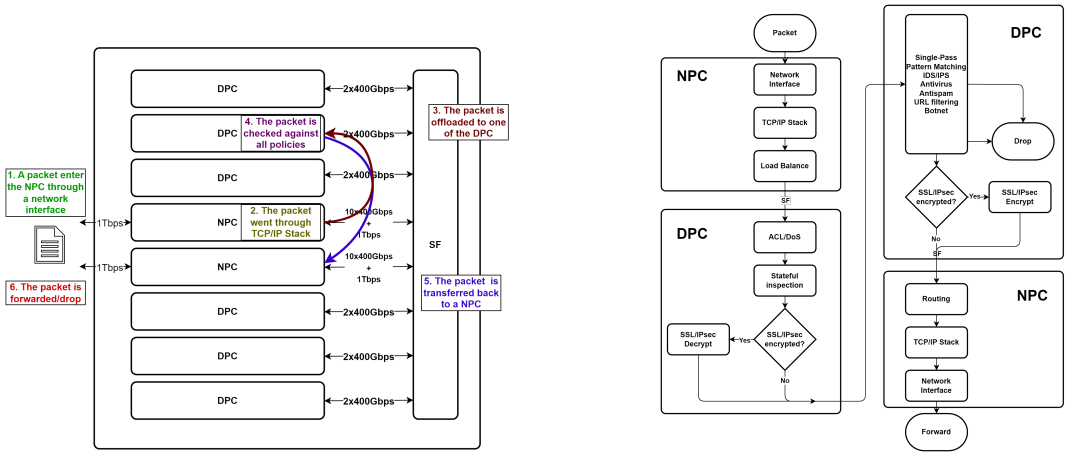

Figure 1. (A) The architecture of a canonical hardware next-generation firewall and (B) the life of a packet through it.

II. THE UTILIZATION BOTTLENECK

A. Heavy-tailed distributions in typical network traffic

Network traffic often demonstrates a heavy-tailed distribution where a small fraction of sessions utilize a large fraction of the available throughput [32]. Figure 2A and Figure 2B show the relationship between session size and the number of sessions in a given size range for each trace. We see (Figure 2A) that the largest flow could be several orders of magnitude larger than the smallest flow. Similarly, we see (Figure 2B) that the top 10% of the flows contribute a large fraction of bytes in almost all the traces. In fact, as little as 0.0017% to 3.71% of the flows could contribute 50% of the overall throughput (we categorize flows into mice and elephant flows using 50% bandwidth division – Table II).

Table II
THE PACKET TRACES USED IN THIS WORK.

Traces	Flow Count	Packet Count	Mice Flows (MFs)	Elephants Flows (EFs)	# Concurrent EFs
CIC-IDS2017	396,968	9,915,680	396,961 [100.00%]	7 [0.00%]	1
MAWI-1	38,625,403	179,844,777	38,622,841 [99.99%]	2,562 [0.01%]	1963
MAWI-2	369,619	1,000,000	368,826 [99.79%]	793 [0.21%]	607
UNIV-1	284,075	19,239,390	284,033 [99.99%]	42 [0.01%]	20
UNIV-2	112,150	98,863,335	112,141 [99.99%]	9 [0.01%]	5
CTU-13-BOTNET	214,925	74,627,232	214,922 [100.00%]	3 [0.00%]	1
CIC-DDOS-1	3,660,401	69,377,252	3,576,130 [97.70%]	84,271 [2.30%]	2
CIC-DDOS-2	11,172,263	55,705,881	10,994,157 [98.41%]	178,106 [1.59%]	3
DARPA-SL-1	836	4,838,519	805 [96.29%]	31 [3.71%]	31
DARPA-SL-2	1,098	5,446,440	1,069 [97.36%]	29 [2.64%]	29
DARPA-SL-3	836	4,841,901	805 [96.29%]	31 [3.71%]	31
DARPA-SL-4	15,850	9,434,732	15,503 [97.81%]	347 [2.19%]	347

B. How elephant and mice flows affect firewall behavior

Elephant flows can cause uneven utilization of DPCs in a hardware firewall. After the packets pass through the NPC for network protocol processing, the load balancer(s) (LB) distribute the packets among a pool of DPCs. Today's load balancers are either implemented in software (using DPDK [56] or ODP [20], for example), or they can be hardware-based load balancers such as the Loadbalancer Enterprise Max [29], shown in Table III. These load balancers support operation in either stateless or stateful mode[1]. In stateless mode, a packet can be forwarded to any worker node. In stateful mode, the load balancer maintains a connection table to route packets from the same flow to the same worker node. Since 1) several firewall functionalities, especially NGFW functionalities, require the state of the session to which the packet belongs in order to process the packet correctly (e.g., stateful inspection) and 2) each DPC stores its own session table that contains all the state information locally for modularity reasons, the LB needs to forward packets that belong to the same session to the same DPC consistently (stateful load balancers). This requirement is also known as *per-connection consistency (PCC)*. The first packet that sets up a session is assigned to a DPC according to the load balancing algorithm implemented by the LB, but the subsequent packets belonging to the session will *always* be assigned to this DPC regardless of the load balancing algorithm used. If we insist on using stateless

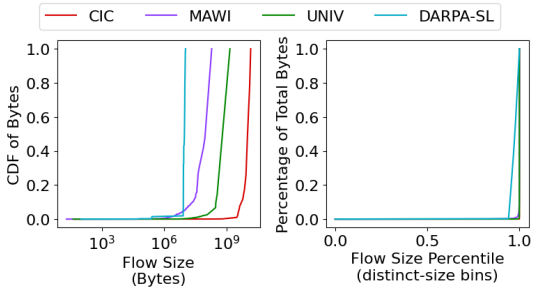

Figure 2. (A) The CDF of flow sizes and (B) the cumulative percentage of flow sizes against the cumulative percentage of the number of flows.

[1]See Section V for how DPDK/ODP/etc. are used.

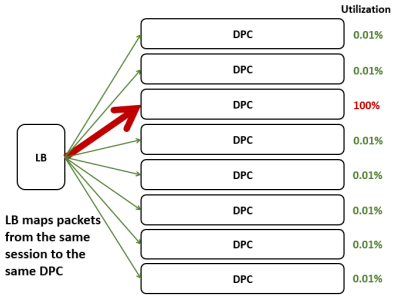

Figure 3. An elephant flow can cause low DPC utilization inside a firewall.

balancers, we need to implement additional mechanisms to distribute the states among the DPCs (see Section V-B).

Table III
COMMON LOAD BALANCER IMPLEMENTATIONS.

Name	Type	Description
DPDK [56]	SW	A set of user-space libraries and drivers that accelerate packet-processing workloads
ODP [20]	SW	APIs for portable high-performance data plane applications
Loadbalancer Enterprise Max [29]	HW	Hardware application delivery controller that supports L4/L7 sticky persistence

PCC load balancing can lead to poor firewall utilization when elephant flows exist. As an illustrative example, consider a case where currently there is a single elephant flow and several mice flow that are transmitting through the firewall (Figure 3). The LB load balances the elephant flow to the third DPC, resulting in 100% utilization of that DPC. The mice flows are load-balanced to the other DPCs. However, since the overall bandwidth of the mice flows is small, they cannot properly utilize the compute resources in DPCs. Therefore, there is a low overall utilization of DPCs inside the firewall.

Firewall utilization would be acceptable if the maximum single-flow throughput were significantly lower than the processing throughput of each DPC. However, we observe that many of today's applications already consume high single-flow bandwidth (relative to single DPC throughput), leading to poor utilization for mid-end firewalls - firewalls with 1-10 DPCs and $\leq$ 10 Gbps/DPC (shown in Figure 5A). The recent advancements in NIC offloading [4] enable significantly higher single-flow throughput – up to 200 Gbps using FPGA-based TCP offloading engines – compared to the maximum of 45 Gbps in existing Linux network stacks shown in [10], further exacerbating the problem for these mid-end firewalls. Future applications could demand single-flow bandwidth up to 1 Tbps [22], [28], [57] (Figure 4). At such single-flow bandwidth values, even the high-end firewalls ($\geq$10 DPCs, $\geq$50 Gbps/DPC) will suffer from poor DPC utilization (shown in Figure 5B). Again, NIC offloading would make the problem even worse. Overall, the firewall becomes a bottleneck in the network path as the elephant flows cannot be processed by multiple DPCs.

We confirm this phenomenon in a realistic setting, where we set up three virtual machines with two VMs (VM0, VM1) running Ubuntu 22.04 and the third VM (VM2) running OPNsense 24.1 [40], acting as the gateway firewall for VM0 (Figure 6A). OPNsense is a popular open-source firewall OS. We enable Suricata [48] IDS + IPS, as well as Zenarmor [54]

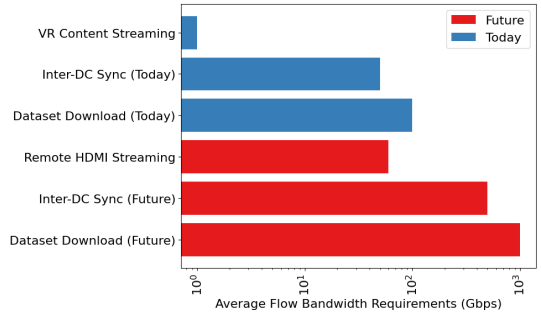

Figure 4. Example applications that demand high single-flow bandwidth.

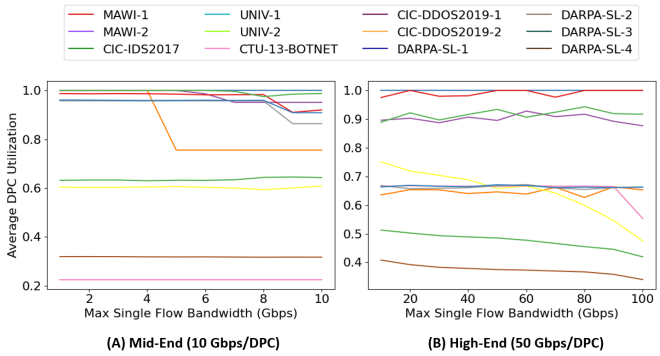

Figure 5. Firewall utilization vs maximum single-flow bandwidth for (A) mid-end and (B) high-end firewalls. The discrepancy between different traces is due to the fact that there are different numbers of elephant flows in the traces.

within OPNsense for NGFW firewall features. We used iPerf3 to generate an elephant flow through the firewall. The average CPU utilization across all cores is recorded as we increase the number of cores allocated to the firewall. Figure 6B shows that multiple cores cannot be efficiently utilized for a single high-throughput session, since the PCC requirement forces the session to be mapped to a single core.

C. Why Session Pinning is Fundamental

Stateful inspection requires co-located state. NGFW packet processing involves maintaining per-session state that is both large (hundreds of bytes to several kilobytes per session, including TCP sequence tracking, reassembly buffers, protocol decoders, and application-layer context) and updated on *every* packet. Distributing this state across DPCs would require either (a) a shared-memory subsystem accessible at line rate from all DPCs, which is infeasible when backplane bandwidth (800 Gbps/DPC) is 4$\times$ lower than on-chip memory

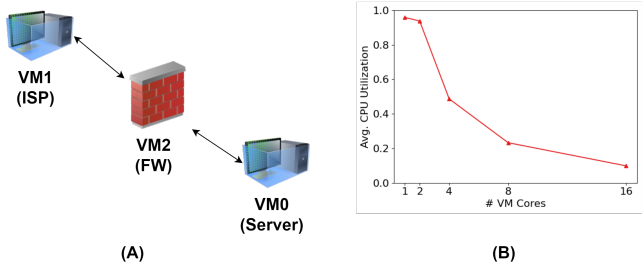

Figure 6. Test environment setup and CPU utilization of VM2 (FW) with various core counts, when running a single TCP session iPerf3 test.

bandwidth (3200 Gbps/PE), or (b) per-packet state migration, which would consume backplane bandwidth proportional to session-state size × packet rate and add multiple microseconds of latency per packet.

Intra-DPC parallelism does not solve inter-DPC imbalance. Modern DPCs already employ multiple processing elements (PEs) internally with a shared session table [19]. This parallelism helps when a single DPC is the bottleneck, but it does not address the root cause: a hash-based or affinity-based session-to-DPC mapping will always suffer from load imbalance given heavy-tailed flow-size distributions. When a 100 Gbps elephant flow is pinned to a 50 Gbps DPC, no amount of intra-DPC threading can process it faster than the DPC's aggregate capacity.

The problem is universal across vendors. Table I shows that every major hardware firewall vendor—Fortinet (FG-7121F, FG-7081F), Palo Alto Networks (PA-7500, PA-7080), and Juniper (SRX5800)—employs a DPC+NPC architecture with PCC-based session pinning. This is not coincidental: stateful, in-order packet processing is a hard requirement of TCP [4], and all vendors converge on the same architectural pattern to satisfy it.

D. Using network traces to quantify impact on utilization

We developed a custom event-driven simulator that replicates the packet flow through a firewall. Our firewall simulator is a cycle–accurate, discrete–event engine that advances in fixed $100\,\mathrm{ns}$ quanta. During each step, the scheduler walks the datapath graph in reverse topological order, beginning with the egress port and finishing at the ingress port, so that back-pressure is resolved within the same simulated cycle. *Modules* within the simulator emulate the behavior of the different subsystems within the firewall (e.g., DPC, NPC, LB, links) as well as the connectivity between these subsystems. The datapath itself is a bidirectional graph. Five LB policies are provided (source-hash, destination-hash, symmetric-hash, round-robin, and least-loaded), while the DPC has two different implementations: a baseline cut-through model and a query-reordering core that prioritizes established flows.

An ingress port injects either synthetic or trace-replayed packets (the frontend can read a packet capture – .pcap file). The packet's record field stores per-module entry and exit timestamps; once the packet leaves the egress port, the simulator derives end-to-end latency directly from these stamps. Each module enforces its internal latency and may hold packets longer when the downstream buffer is full, thereby modelling head-of-line blocking.

Ingress traffic can be throttled by an additive-increase-multiplicative-decrease (AIMD) controller that halves the offered load whenever the ingress queue exceeds a threshold and grows linearly otherwise. Two traffic-engineering optimisations are supported. Elephant-flow skipping compresses headers of heavy-hitter connections, while TinyFlow randomises TCP/UDP ports of elephants so that hash-based LBs distribute their packets more evenly.

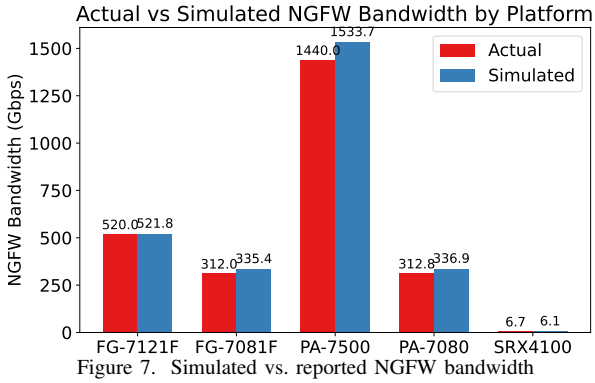

Figure 7. Simulated vs. reported NGFW bandwidth

Previous work has employed similar event-driven approaches for high-throughput network simulation [39] [25].

To validate our simulator, we model a firewall topology closely matching the FG-7121F, a modern hardware NGFW composed of 2 NPCs (including load-balancing functionality) and 10 DPCs, with a reported aggregate NGFW throughput of 520 Gbps. We configure the simulator using component-level parameters derived from the FG-7121F specification: each DPC is assigned 52 Gbps throughput, consistent with the per-DPC throughput implied by the datasheet [16], and each NPC is configured at 1 Tbps to reflect its packet-distribution bandwidth. To minimize artificial throttling from undersized queues, the buffer capacity of each module is set to 1.5 × throughput × latency, which is sufficient to absorb transient congestion while preserving realistic queueing effects. We further implement the default PCC load-balancing algorithm used in FG-7121F, namely symmetric hash [18], at each NPC so that flow-to-DPC mapping matches the hardware design.

Using this configuration, we run the simulator on randomly generated traffic with uniformly distributed source and destination IP addresses and measure the maximum sustainable throughput. The simulator achieves 510 Gbps, closely matching the reported 520 Gbps throughput of the FG-7121F. The small gap suggests that the model faithfully captures the dominant performance constraints of the real system, including load distribution, per-module service capacity, and buffering effects. Since other hardware firewalls use the same basic architecture and differ primarily in component counts and specifications, the same simulation framework can be easily retargeted to the additional commercial NGFWs listed in Table I. Figure 7 shows that the simulated throughput values align closely with the reported bandwidths for these systems as well, providing further evidence of simulator fidelity.

We used the simulator to quantify the impact of elephant and mice flows on the utilization of hardware firewalls. Simulations were performed for different network traces on a 10-DPC, 2-NPC, and 1-LB hardware firewall. We perform the evaluations for three widely used load-balancing algorithms – round robin (RR) where sessions are distributed to the DPCs in rotating order, least utilized (LU) where a session is allocated to the DPC with currently the lowest utilization, and symmetric hashing (SH) where a session is allocated to the DPC with id $= hash(IP_A, Port_A, IP_B, Port_B)$ mod N, where N is the

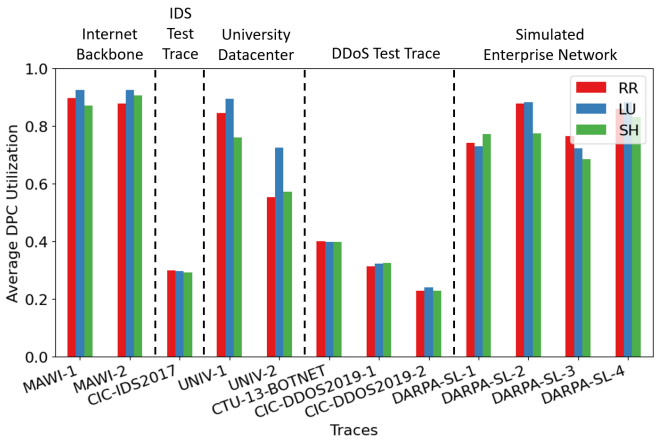

Figure 8. The average utilization of DPCs when using various load-balancing algorithms, with a 10-DPC, 2-NPC, 1-LB setup.

number of NPCs in total. Due to the PCC restriction discussed earlier, all load balancers work at the session level. LB makes a decision for the first packet in a session, and later packets are forwarded to the same DPC.

Figure 8 shows the results. For MAWI-1:2, UNIV-1:2, DARPA-1:4, we see an average DPC utilization of 90%, 90.9%, 84.5%, 64.8%, 74.7%, 84.5%, 72.4%, and 85.7% respectively. MAWI-1:2 traces are captured at the Internet backbone and consist of a large number of concurrent sessions. This leads to relatively high utilization. At the other extreme, CIC-IDS2017, CTU-13-BOTNET, and CIC-DDOS2019-1:2 consist of fewer elephant flows and a large number of mice flows and, therefore, have poor utilization. Although the utilization is relatively high for half of the traces with 10 DPCs, we will show in the next sub-section that as we keep scaling the number of DPCs and the throughput per DPC, the utilization significantly decreases even for these traces because of load imbalance. We also observe that utilization is only slightly dependent on the specific load-balancing algorithm. LU shows a slightly higher utilization than RR and SH, as it explicitly considers the DPC utilization information.

E. Utilization plummets as the number of DPCs scales

To make matters worse, firewall utilization decreases as the number of DPCs increases. In essence, additional DPCs do not help process elephant flows, since packets from the same session continue to be mapped to the same DPC, resulting in reduced overall utilization with every additional DPC. Figure 9 shows how throughput scales when the number of DPCs is swept from 1 to 10 for mid-end firewalls and from 10 to 50 for high-end firewalls while maintaining a fixed ratio of 1 LB: 1 NPC: 5 DPCs in the simulation. We present results for various maximum single-flow bandwidth values that represent both realistic and forward-looking network traffic scenarios (10 Gbps / 5 Gbps / 1 Gbps for mid-end firewalls, 100 Gbps / 60 Gbps / 10 Gbps for high-end firewalls). The DPC processing bandwidth is selected to be 10 Gbps for mid-end firewall and 50 Gbps for high-end firewall results. The backplane bandwidth is set to 19.2 Tbps which is common

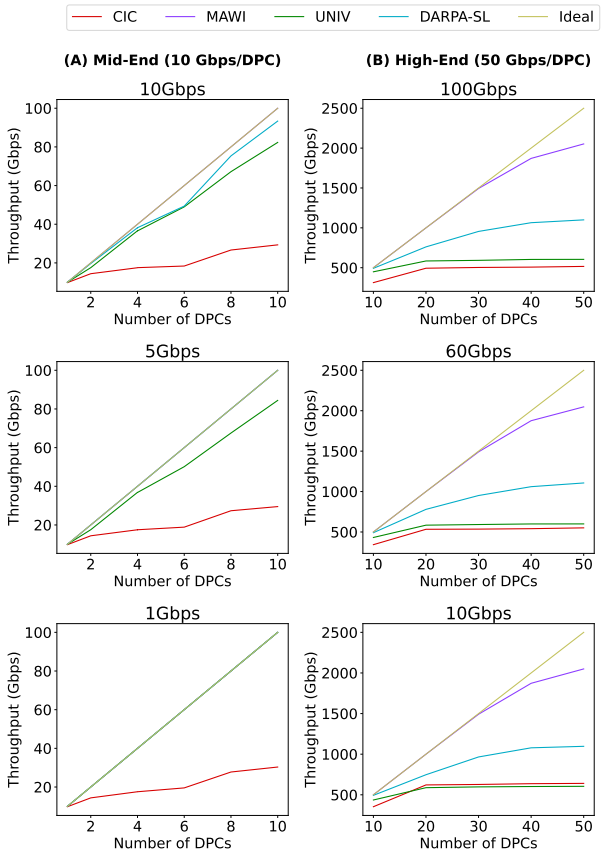

Figure 9. Throughput vs the number of DPCs for both mid-end and high-end firewalls, at various maximum single-flow bandwidths. The backplane bandwidth is set to 19.2 Tbps.

among hardware firewalls [16], [37] (the same bandwidth provided by a Nexus 9800 fabric modules [11])

The results show that the firewall throughput is much lower than the ideal throughput (the compute throughput of a single DPC - multiplied by the number of DPCs). This is true even for infinite backplane bandwidth where the gap is explained entirely by the low utilization produced by the heavy-tailed distribution of the network traces. Second, the gap between the ideal throughput and the observed throughput keeps increasing with the number of DPCs, even for the infinite bandwidth case. This, again, is attributed to the worsening utilization. Finally, the throughput difference between ideal and infinite backplane bandwidth is much bigger than the throughput difference between various backplane bandwidth values. This shows that the overall achieved throughput of a hardware firewall is much more sensitive to poor utilization caused by the PCC constraints rather than the available backplane bandwidth.

III. CCPA: Uncovering Parallelism in Same-Session Packet Processing

We observe that while stateful inspection requires sequential consistency, content inspection can be performed in parallel without impacting correctness. Content inspection takes the packets, the policies, and the session state as input and es-

sentially performs pattern matching on millions of signatures. Some of the signatures are from a known virus, malware, exploit, etc. During this stage, some firewalls also employ machine learning models to detect unknown threats. There is no fundamental reason why content inspection for different packets belonging to the same session cannot be performed in parallel.

We leverage the above novel observation to advocate serial processing of stateful inspection (on the same DPC) and concurrent processing of content inspection (across multiple DPCs) for packets belonging to the same session. Figure 10 presents a simple illustration of the idea. The performance benefits would be determined by p, the ratio of time spent in stateful inspection vs. content inspection of a packet.

To estimate an example realistic upper bound on p (and, therefore, realistic lower bound on the benefits of parallelizing content inspection), we timed (Table IV) an open source kernel representing stateful inspection (NPF [43] is a stateful layer 3 packet filter) and two open source kernels representing computation during content inspection (Suricata [48] performs IDS/IPS, nDPI [3] implements application identification during DPI) to calculate the processing time per packet for each kernel on every available network trace. If we assume that App-ID and IDS are the only two kernels in the content inspection stage (unlikely, since modern firewalls execute many more kernels during content inspection) and that the content inspection kernels follow stateful inspection during firewall execution, we see a p value of only 0.45% (0.0045). For this realistic upper bound value of p, we can get a 40x speedup at 50 DPCs and a 10x speedup at 10 DPCs.

Table IV
PROCESSING TIME PER PACKET FOR DIFFERENT KERNELS, AT DEFAULT, MINIMUM (LEAST SECURED, FASTEST), AND MAXIMUM (MOST SECURED, SLOWEST) FIREWALL SETTINGS.

Kernels	Default (µs per packet)	Minimum	Maximum
Stateful Inspection (NPF [43])	2.15	1.97	2.15
App-ID (nDPI [3])	221	19	309
IDS (Suricata [48])	253	0	872
p (Stateful Inspection portion)	0.45%	10.4%	0.18%

We call our novel architectural approach - *CCPA*. In this **C**oncurrent **C**ontent **P**rocessing **A**rchitecture. One way to implement CCPA is to let DPCs perform state checking. After a DPC finishes stateful inspection for a packet, it load balances the packet to one of the DPCs, including possibly itself, for parallel content inspection. However, this creates a need for all-to-all communication among the DPCs, stressing the requirement of the backplane. For example, if we have n DPCs and each DPC runs at 50 Gbps, the backplane now needs to have an additional radix-n, 320 Gbps crossbar in order to support sending both the states and the packets between DPCs. This could also potentially increase the port-to-port latency as the packets need to take an additional hop from one DPC to another. In the worst case, we estimate that the additional hop between two DPCs could introduce 640 ns of additional latency (switching: 200 ns, propagation through the backplane: 210 ns, transmission of 9.5 KB of state, and packets: 230 ns). Typical firewalls have a port-to-port latency of 7.5 µs). These disadvantages led us to choose a different implementation.

To implement CCPA efficiently, we instead introduce a new module called State Checking Card (SCC) (Figure 11) that only handles stateful inspections. SCC contains the session table and computing resources required for performing stateful inspection. As soon as a packet finishes stateful inspection, the SCC sends the packets along with the updated session state that it just computed to a DPC. The DPC can then perform content inspection without communications with other DPCs. This is because of the fact that content inspection is compute-intensive and stateless (e.g., signature matching, ML model pass). Since stateful inspection for packets belonging to the same session is performed on a single chip, throughput could be further optimized when dealing with a continuous packet stream from an elephant flow. Since each chip's session state is mutually exclusive, the write updates of the state to the slower DRAM can be delayed via write-back caches. In addition, no additional fetches are needed between back-to-back stateful inspections, as the updated state is just computed.

Because the SCC's per-packet work is intentionally minimal—a session-table lookup and a dispatch decision, with all heavy content inspection offloaded to the DPCs—SCC bottlenecks will be rare in practice. For small-packet-dominated workloads, the SCC can be scaled further via parallel pipelines or multiple hash-partitioned instances.

Elephant flows also do not starve mice flows at the SCC. Since the SCC operates at session granularity—one lookup per session establishment, with no further lookups for subsequent packets—and elephant flows are rare by count (0.001%–3.7% of flows) despite being large by payload, they consume a negligible share of SCC lookup capacity. Fairness can be further enforced via per-class queues with weighted service or by bounding per-flow outstanding packets at the SCC input.

Figure 12 shows the overall architecture. Packets arrive at the network interfaces on the SCC card. Within each SCC, 4 state processors (SP) are used for stateful inspection. We use a stateful load balancer to map the incoming packets to the SPs. I.e., the load balancer ensures that packets belonging to the same session are mapped to the same SP. This eliminates the need to support shared memory across the SPs. Once a packet is mapped to an SP, the SP performs stateful inspection. For performing inspection, each SP has a private memory capacity of 2.56 TB (10 256GB-DDR5 DIMMs) for storing its session table and incoming packets (this value can be customized depending on the usecase of the firewall. Here we assume 2.56 TB in order to match the capability of state-of-the-art hardware NGFWs [37]). Each SP performs stateful inspections at 1.2 Tbps and forwards the checked packet and the corresponding state through a crossbar switch to one of the DPCs. We currently assume a round-robin policy for assigning checked packets to the DPCs. Since the content inspection throughput of a DPC is 50 Gbps and session state (8 KB) will be forwarded by an SP to a DPC for every packet (1500 bytes average [51]) being content inspected by the DPC, we estimate that only $50 * (8000 + 1500)/1500 = 330$ Gbps of backplane bandwidth per DPC is needed to saturate the available compute resource in each DPC (50 Gbps * the bandwidth ratio increase

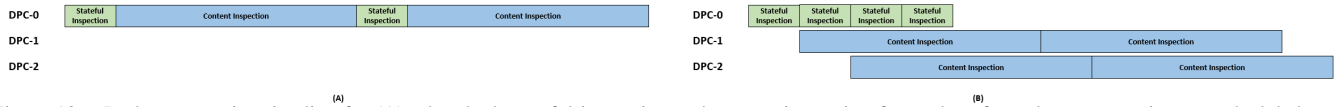

Figure 10. Packet processing timeline for (A) when both stateful inspection and content inspection for packets from the same session are scheduled on the same DPC and (B) when content inspection can be performed on other DPCs.

due to sending the session state). If jumbo frames [7] are used, the required backplane bandwidth would be reduced as the packets per second reduced for the same throughput. Although the SP forwards the packets to DPCs, it will still keep the packet in its 2.56 TB of memory. As each DPC finishes the content inspection and a decision is generated, the decision, instead of the entire packet is sent back to the SP, saving backplane bandwidth and reducing latency. It is only when the packet is forwarded/dropped that the memory holding the packet is freed. One additional factor we need to consider is that while content inspection can go in parallel, packets must be sent out of the firewall in order. If packets are sent out of order by more than 3 packets, TCP protocol on the sender side will ask for retransmission for those out-of-order packets, causing throughput degradation. To ensure in-order transmission of packets, we allocate a dynamic in-memory buffer (*packet reorder buffer*, pROB) to hold all the packets when they enter the firewall and forward a packet after content inspection only if the previous packet has been forwarded. This buffering of packets also eliminates the need to transmit packets from DPC back to NPC in the baseline architecture (since only the content inspection decision can now be communicated to the NPC that already has the packet buffered), reducing the required backplane bandwidth by 50%. For a given per-flow fan-out F (i.e., a single flow's packets can be content-inspected on F DPCs in parallel), we size the packet ReOrder Buffer (pROB) needed at the SCC to ensure in-order egress. The worst-case pROB occupancy occurs when decisions for later packets return before decisions for earlier packets; the SCC must hold all "early" packets until the missing earlier decision arrives. The maximum needed pROB capacity is shown in Figure 13. As the number of DPCs scales to 50, the maximum pROB capacity for a high-end firewall with 4.8 Tbps at the SCC and 50 Gbps per DPC is <60 MB. CCPA adds one additional logical stage (SCC before DPC content inspection), and may introduce reorder buffering at egress (pROB). The worst-case added hop latency is bounded to be 440 ns (propagation through the backplane: 210 ns, transmission of packets and 9.5 KB of state: 230 ns), which is small (≈3%) compared to typical firewall budgets (7.5 µs).

Several practical concerns arise with pROB-based egress reordering. In normal operation, all packets within a session are forwarded to the same DPC, so intra-session order is naturally preserved and the pROB introduces zero reordering; cross-session reordering is irrelevant since TCP treats each connection independently. If a packet or its inspection result is lost, strict in-order egress could stall subsequent packets behind a hole. We prevent this without global synchronization using timeouts (treat missing packets as dropped after a config-

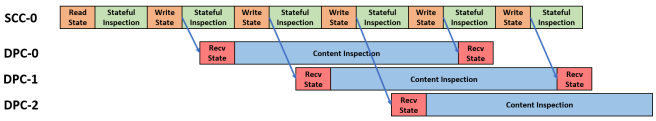

Figure 11. The packet processing timeline of CCPA.

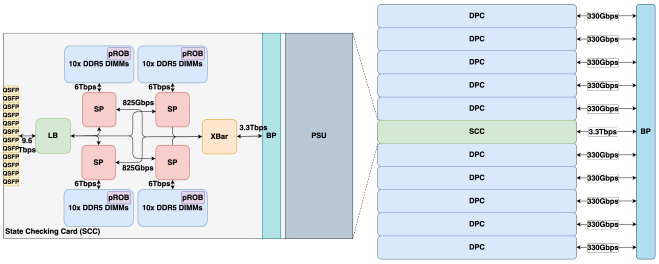

Figure 12. A CCPA firewall.

urable deadline), bounded retries, and hole-skipping policies. The pROB is organized per session so that a hole in one flow does not stall unrelated sessions.

Credit-based flow control between SCC and DPCs throttles injection when a DPC's return path is impaired: when credits are exhausted for a given DPC, only that DPC's packets are buffered while all others continue, providing per-DPC stall isolation and preventing congestion collapse. pROB bottlenecks can arise under high fan-out combined with high variability in content-inspection completion times, but the combination of credit-based flow control, per-session organization, and timeout-triggered hole-skipping keeps occupancy bounded.

Some challenging patterns (such as scanning for virus signatures in downloading files) may require scanning data that spans multiple packets[2]. In such cases, CCPA needs to operate slightly differently. First, the SP will detect that a particular flow matches the application ID for multi-packet inspection based on stateful inspection on the first few packets. If a particular flow requires multi-packet inspection, the SP would start storing the entire flow until the flow ended – unlike the normal inspection where stored packets are discarded after

[2]Multi-packet inspection is a high overhead feature; Most hardware NGFWs do not enable this feature by default [19], others do not support it altogether [34]

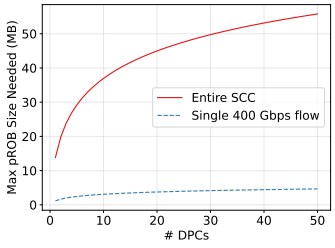

Figure 13. The maximum size of pROB needed.

forwarding. For every new packet received, the SP will send all cached packets within the flow to a DPC. The DPC can then view the entire flow history during content inspection, allowing useful signature identification.

Also, parallel content inspection forbids any algorithm that requires updating the session state in an in-order state update semantics after the content inspection is complete. Fortunately, for almost all pattern-matching algorithms that exist in firewalls, this requirement does not exist [1]. For flows that require sequential consistency during content inspection, one candidate solution is to fall back to Per-Connection-Consistency processing. PCC fallback is rare. We audited the ET Open Suricata 7.0.0 ruleset and found that only 5.42% rules require PCC fallback. The remaining 94.6% of rules run fully in parallel under CCPA.

IV. Prototyping the State Processor

We used AMD Vitis HLS to implement a state processor (SP) that performs the typical NGFW state inspection tasks [3] and mapped it to a Zynq-7000 Xilinx FPGA. Our SP design occupies 5 % LUTs, and closes at >400 MHz, sustaining 1 Gbps at minimum packet size. We tested othe FPGA implementation with a simulation testbench that continuously generates random packets. The packets are sent into the DUT through the AXI bus, where they are processed and then load-balanced to a mock DPC. The mock DPC returns a flag to the DUT, and the DUT forwards the packet to the output port. We ran the testbench and did not encounter any packet reordering or packet drops when the packet was sent into the DUT at 1 Gbps with 1 KB packets.

To further validate the CCPA architecture beyond simulation, we constructed an end-to-end prototype combining our FPGA-based SP with a software-based DPC running Suricata [44] for content inspection. In this experiment, the FPGA SP performs stateful inspection on live pcap-replayed traffic (MAWI-2 trace), then forwards checked packets over an AXI-to-Ethernet bridge to a commodity server running Suricata. The server returns content inspection decisions to the SP, which performs pROB-based reordering and forwards the packets. This prototype demonstrates that the CCPA pipeline – stateful inspection on dedicated hardware followed by content inspection on separate compute – functions correctly end-to-end: no packets were reordered or dropped during a 30-minute run at sustained 188 Mbps throughput. While the prototype's throughput is limited by the single-server DPC and the unoptimized DPC software stack, it validates the architectural decomposition and the pROB mechanism in a real hardware/software environment.

V. Quantifying CCPA benefits

We modeled CCPA using our simulator described in Section II) (Table V lists the parameters used for simulations). The values correspond to components of a typical hardware NGFW. The throughput achieved for CCPA for 1-10 DPCs (mid-end firewalls [15]) and 10-50 DPCs (high-end firewalls [16]) for all traces is shown in Figure 14. The results are presented for the round-robin load balancing algorithm. The results are within 5% when *least utilized* load balancing algorithm is used (not shown).

Table V
Simulation Parameters.

Components	Throughput (Gbps)	Latency (ns)	Power (W)	Volume (RU)
DPC (mid-end) [15]	10	3,000	280	1.5
DPC (high-end) [16]	50	3,000	675	1.5
NPC [16]	1,000	300	568	1.5
SCC	4,800	300	1,000	2
TH5 [2]	51,200	200	1,500	2

We see that CCPA outperforms the baseline architecture on high-end firewalls by 1.22x for internet backbone traffic (MAWI), by 4.07x for university data center traces (UNIV), by 1.4x for enterprise network (DARPA-SL), and by up to 1.69x for other traces (CIC-DDOS2019, CIC-IDS2017, CTU-13-BOTNET) when the number of DPCs is small (<30). The benefits increase with the number of DPCs, to 1.2x for MAWI, 11.05x for UNIV, 2.43x for DARPA-SL, and up to 2.13x for other traces. Depending on the number of elephant flows and the number of concurrent elephant flows within a trace, the normalized benefits of CCPA over baseline firewall varies. Due to the large amount of flows in MAWI, the baseline utilization is already high (Figure 8). UNIV and CIC contain a small number of concurrent elephant flows, leading to load imbalance in the baseline firewall as the number of DPCs scales. Therefore, they demonstrate higher benefits when using CCPA at high number of DPCs. On average across all traces, we see a 3.46x increase in throughput with CCPA when we consider a realistic maximum single-flow bandwidth is 10 Gbps. As we project into the future, we see even more benefits of using CCPA: a 4.29x increase in throughput on average across all traces.

Mid-end firewalls see lower gains. Since mid-end firewalls have fewer DPCs, the baseline architecture is able to achieve higher utilization compared to high-end firewalls even with elephant flows, leading to lower benefits from CCPA. Despite this, we still achieve up to 1.45x throughput improvement over the baseline on the UNIV network trace. On average, we see an improvement of 1.08x.

We also compare CCPA against alternative approaches to improve the utilization of a hardware firewall. Several alternatives are possible. For example, elephant flows can be broken into mice flows [52], elephant flows' throughput [12] can be throttled, content inspections can be skipped for elephant flows entirely [33], elephant flows can still be distributed across multiple sessions and state can be exchanged between the DPCs frequently [35]. We evaluated three alternative approaches – using a stateful load balancer and breaking elephant flows into mice flows (Stateful LB + TinyFlow), using a stateless load balancer and exchanging states across the backplane (Stateless LB + State Exchange), as well as using a stateful load balancer and skipping elephant flow inspection (Stateful LB + Skip EF)) – in our simulator for comparison with our proposed architecture at 10/60/100 Gbps of maximum single-flow bandwidth. In practice, these approaches could be implemented with DPDK/ODP, or directly in hardware.

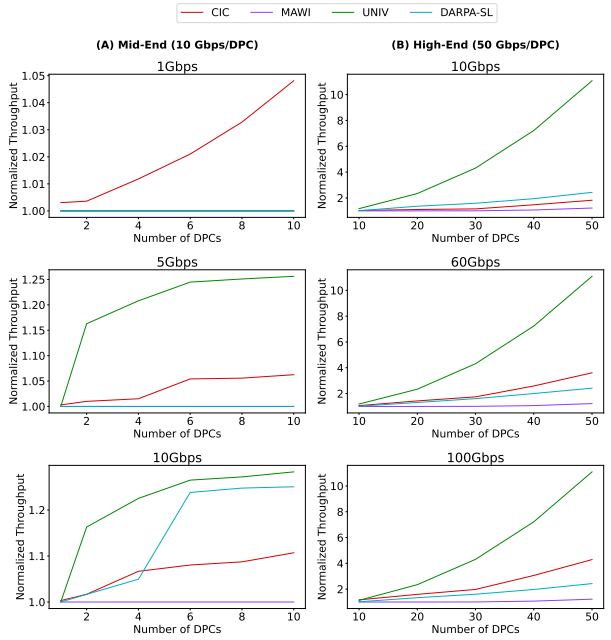

Figure 14. Normalized throughput of CCPA, for mid-end and high-end firewalls for different maximum single-flow bandwidths. The backplane bandwidth is set to 19.2 Tbps.

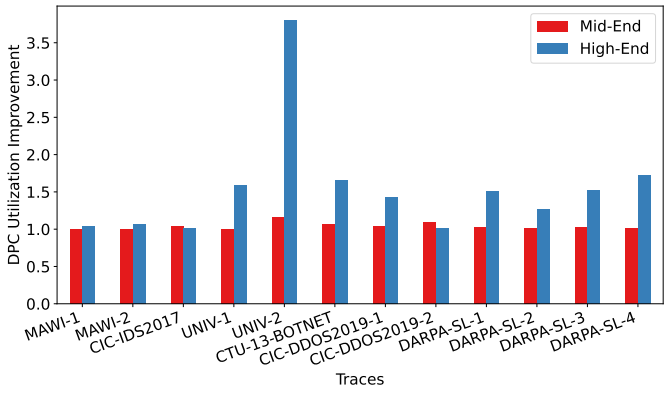

Figure 15. Utilization improvement of CCPA over baseline for mid-end and high-end firewall for 10 DPCs.

The results (Figure 16) show that TinyFlow, state exchange, and skip EF achieve up to 1.84x, 3.16x, and 3.7x increase in firewall throughput averaging across all traces (vs 4.29 for CCPA) at 100 Gbps of maximum single-flow bandwidth. The results for other maximum single-flow bandwidths are similar. Although elephant flows are broken up into smaller mice flows in TinyFlow, a heavy-tailed distribution still exists, since we break up the flows only at the session level, but these tiny flows can still potentially create uneven utilization among DPCs since the flow-to-DPC mapping is still static, limiting the improvement from employing TinyFlow. State exchange, on the other hand, demands higher backplane bandwidth (330 Gbps * N DPCs additional bandwidth) as well as a complex locking mechanism among the DPCs to ensure state correctness. Therefore it performs worse than CCPA at the same 19.2 Tbps backplane bandwidth. Skip EF performs close to the state-exchange optimization. However, although content

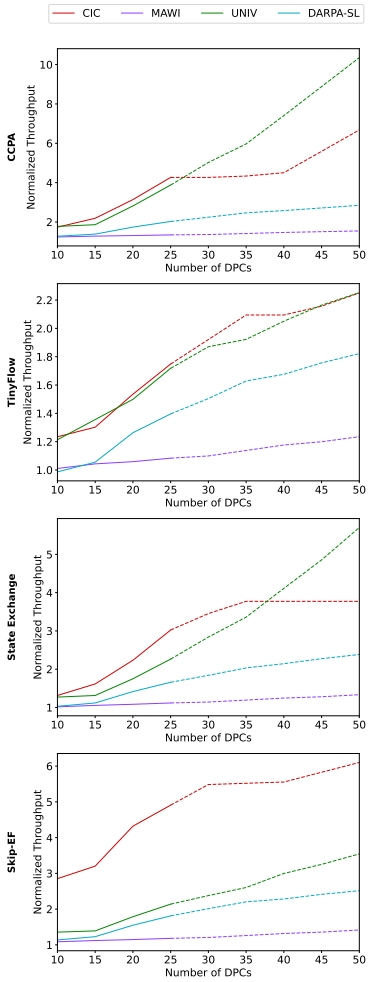

Figure 16. Normalized throughput (over baseline) of alternative solutions compared to CCPA for a 19.2 Tbps backplane bandwidth.

inspection of elephant flows is skipped, uneven distribution of *sub-elephant flows* (flows with throughput between mice flows and elephant flows) will still affect the throughput achieved by Skip EF.

We also evaluated throughput for a shared state firewall, where all DPCs access a centralized shared session table. This makes DPC allocation no longer session persistent, addressing the DPC utilization problem. The result shows that the shared state architecture degrades NGFW throughput by 9.1x when assuming the same backplane bandwidth, since the backplane bandwidth (800 Gbps / DPC) is not sufficient for supporting shared memory session table fetches.

We also studied sensitivity to the backplane architecture. While our 19.2 Tbps backplane was sufficient for the firewalls studied earlier, conventional backplane technologies may be unable to support the 6.33x increased bandwidth needed for CCPA as the firewall throughput scales in future. We evaluate CCPA for much higher throughput firewalls for three novel backplane technologies listed in Table VI. The Nexus 9800 fabric module supporting 19.2 Tbps is considered our baseline and was assumed as the backplane for all our prior results. It is mounted vertically at the back of the chassis, connecting all

Table VI
DIFFERENT BACKPLANE TECHNOLOGIES AND THEIR RESPECTIVE CHARACTERISTICS.

Technology	Bandwidth (Tbps)	Energy (pJ/b)	Description
Nexus 9800 Fabric Module [11]	19.2	8	Fabric modules that are mounted vertically on the back, connecting all components together.
ExaMAX Backplane [44]	43	5	PCIe-based backplane solutions with custom electrical cablings
Optical Backplane [8]	>100	2	On-board optical/electrical converted, connected with optical fibers

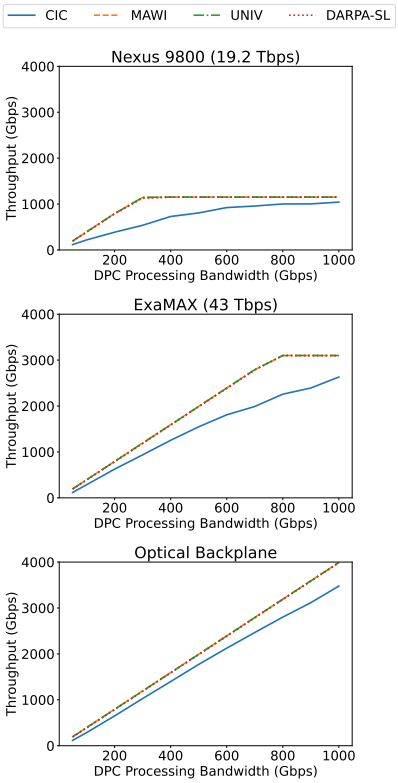

Figure 17. Throughput of CCPA across different backplane technologies.

the DPCs/NPCs/LBs together. Another solution, the ExaMAX backplane system supporting 43 Tbps, provides high-density connectors and a cabling system for board-to-board PCIe communications. Besides conventional electrical connections, we also consider the use of optical backplanes. Optical links can provide significantly higher bandwidth density compared to electrical links by leveraging wavelength-division multiplexing (WDM) while consuming significantly less energy per bit. Prior work has achieved 12.5 Tbps per fiber core [8], which can support DPCs with 40x the processing bandwidth of today.

Figure 17 shows how the performance of CCPA depends on the choice of the backplane architecture for very high throughput firewalls, for 10 DPCs and 50 Gbps to 1 Tbps per DPC. We see that an optical backplane allows CCPA to achieve near-optimal throughput, a 3.3x improvement over the baseline backplane technology. We also estimate that the total energy is only increased by 30% compared to the baseline despite the significant increase in backplane throughput. This is due to the much higher energy efficiency of optical links vs electrical links (Table VI).

CCPA does incur an additional power and cost (Table VII). But a 4.29x higher throughput on real-world traces (Section V) may provide an acceptable tradeoff.

Additional Component	Cost ($)	Power (W)
7.5 TB DDR5	51540	750
SCC	≈20000	≈600
% increase	3.35	16.46

Table VII
CCPA OVERHEAD VS PA-7500.

VI. RELATED WORK

Load balancing, PCC, and heavy-tailed traffic Representative PCC designs span consistent-hashing software LBs (e.g., Google's Maglev [14] which supports consistent hashing and connection tracking, explicitly preserving PCC) and fabric-aware flowlet scheduling (e.g., CONGA [6] which distributes flowlets by congestion feedback across a Clos fabric). CCPA preserves PCC for the stateful phase only, then decouples and parallelizes content inspection across DPCs to improve DPC utilization in the presence of elephant flows.

Split-phase packet processing and SmartNIC offload FlexTOE [45] shows that separating TCP processing into a stateful component and a stateless, fine-grained parallel component on SmartNICs unlocks concurrency while preserving semantics. CCPA adopts the same high-level principle but at NGFW granularity: (i) we keep PCC and ordering within a serialized stateful inspection stage; (ii) we then fan-out content inspection across DPCs; and (iii) we analyze system-level scaling (backplane demand, multi-chassis deployment) beyond a single NIC or host pipeline.

DPI and content-inspection acceleration Classic approaches include Aho–Corasick variants (bit-split engines) [5], D2FA-style DFA compression [26], and more recent hybrids like Hyperscan that mix DFA/NFA engines with SIMD acceleration [50]. System-level designs leverage CPUs/GPUs/FPGAs: Kargus [23] exploits batching and GPU offload to reach tens of Gbps; Pigasus [55] attains 100 Gbps on a single FPGA-equipped server with careful memory/algorithm design. CCPA is orthogonal to DPI engine choice: any faster kernel lowers per-packet cost, and CCPA further scales throughput by enabling intra-flow parallel content inspection across DPCs after a single stateful pass.

VII. CONCLUSION

In this work, we showed that the architecture of today's hardware firewalls limits their utilization for network traffic dominated by elephant flows. We made a novel observation that the content inspection phase of packet processing, which dominates the overall processing time, can be parallelized across multiple DPCs without impacting correctness. We leveraged this observation to propose CCPA, a novel architecture in which the stateful inspection for all packets in a session is first performed sequentially on a dedicated processor before the packets are sent to DPCs for distributed content inspection. We showed that a hardware implementation of CCPA improves the average firewall throughput by 4.29x (14.4x when using an optical backplane).

REFERENCES

[1] [Online]. Available: https://docs.paloaltonetworks.com/advanced-threat-prevention/administration/threat-prevention/threat-signature-categories

[2] "Tomahawk 5 - ethernet network switches." [Online]. Available: https://www.broadcom.com/products/ethernet-connectivity/switching/strataxgs/bcm78900-series

[3] "ntop/ndpi," Apr 2024. [Online]. Available: https://github.com/ntop/nDPI

[4] 2025. [Online]. Available: https://www.linkedin.com/pulse/pushing-limits-200g-ethernet-multi-stream-data-toe200gadv-ip-iybfc/

[5] A. V. Aho and M. J. Corasick, "Efficient string matching: An aid to bibliographic search," *Communications of the ACM*, vol. 18, no. 6, pp. 333–340, 1975.

[6] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav, and G. Varghese, "Conga: Distributed congestion-aware load balancing for datacenters," in *Proceedings of the 2014 ACM SIGCOMM Conference*, Chicago, IL, USA, 2014.

[7] E. Alliance and B. Kohl, "Ethernet jumbo frames," 2009.

[8] N. Bamiedakis, K. A. Williams, R. V. Penty, and I. H. White, "Chapter 11 - integrated and hybrid photonics for high-performance interconnects," in *Optical Fiber Telecommunications (Sixth Edition)*, sixth edition ed., ser. Optics and Photonics, I. P. Kaminow, T. Li, and A. E. Willner, Eds. Boston: Academic Press, 2013, pp. 377–418. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123969583000111

[9] V. Bollapragada, M. Khalid, and S. Wainner, *IPSec VPN design*. Cisco Press, 2005.

[10] Q. Cai, S. Chaudhary, M. Vuppalapati, J. Hwang, and R. Agarwal, "Understanding host network stack overheads," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, ser. SIGCOMM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 65–77. [Online]. Available: https://doi.org/10.1145/3452296.3472888

[11] Cisco, "Cisco nexus 9800 series switches data sheet." [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/nexus9800-series-switches-ds.html#Productspecifications

[12] Cisco, "Cisco secure firewall management center device configuration guide, 7.2 - elephant flow detection [cisco secure firewall management center]." [Online]. Available: https://www.cisco.com/c/en/us/td/docs/security/secure-firewall/management-center/device-config/720/management-center-device-config-72/elephant-flow.html

[13] Cloudflare, "What is a next-generation firewall (ngfw)?" [Online]. Available: https://www.cloudflare.com/learning/security/what-is-next-generation-firewall-ngfw/

[14] D. E. Eisenbud, C. Yi, C. Contavalli, C. Smith, R. Kononov, E. Mann-Hielscher, A. Cilingiroglu, B. Cheyney, W. Shang, and J. D. Hosein, "Maglev: A fast and reliable software network load balancer," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. Santa Clara, CA: USENIX Association, Mar. 2016, pp. 523–535. [Online]. Available: https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/eisenbud

[15] Fortinet, *FortiGate 7000E Series FG-7060E, FG-7040E, and FG-7030E*. [Online]. Available: https://www.fortinet.com/content/dam/fortinet/assets/data-sheets/FortiGate_7000_Series_Bundle.pdf

[16] Fortinet, *FortiGate 7000F Series FG-7121F, FG-7081F, FG-7081F-2, FIM-7921F, FIM-7941F, and FPM-7620F*. [Online]. Available: https://www.fortinet.com/content/dam/fortinet/assets/data-sheets/fortigate-7000f-series.pdf

[17] Fortinet, "About session-aware load balanced clusters (slbcs) — session-aware load balancing cluster guide," 2024. [Online]. Available: https://docs.fortinet.com/document/forticontroller/5.2.11/session-aware-load-balancing-cluster-guide/855962/about-session-aware-load-balanced-clusters-slbcs

[18] Fortinet, "Changing the load distribution method — session-aware load balancing cluster guide," 2024. [Online]. Available: https://docs.fortinet.com/document/forticontroller/5.2.11/session-aware-load-balancing-cluster-guide/606276/changing-the-load-distribution-method

[19] Fortinet, "Fortigate-7121f fast path architecture — hardware acceleration," 2024. [Online]. Available: https://docs.fortinet.com/document/fortigate/7.0.13/hardware-acceleration/55666/fortigate-7121f-fast-path-architecture

[20] O. Foundation, "Technical overview – opendataplane." [Online]. Available: https://opendataplane.org/index.php/services/technical-overview/

[21] L. Guo and I. Matta, "The war between mice and elephants," in *Proceedings Ninth International Conference on Network Protocols. ICNP 2001*, 2001, pp. 180–188.

[22] M. Hoagland, "Bandwidth requirements for real-time replication — sios," May 2021. [Online]. Available: https://us.sios.com/blog/how-much-bandwidth-do-you-need-to-support-real-time-replication/

[23] M. Jamshed, J. Lee, S. Moon, I. Yun, D. Kim, S. Lee, Y. Yi, and K. Park, "Kargus: A highly-scalable software-based intrusion detection system," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS)*, 2012, pp. 317–328.

[24] Juniper, *SRX5400, SRX5600, SRX5800 FIREWALLS DATASHEET Product Description*. [Online]. Available: https://www.juniper.net/content/dam/www/assets/datasheets/us/en/security/srx5400-srx5600-srx5800-firewall-datasheet.pdf

[25] R. Kirsch, "ryankgit/packet-firewall-simulation," Mar 2021. [Online]. Available: https://github.com/ryankgit/packet-firewall-simulation

[26] S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley, and J. S. Turner, "Algorithms to accelerate multiple regular expressions matching for deep packet inspection," in *Proceedings of the 2006 ACM SIGCOMM Conference*, Pisa, Italy, 2006, pp. 339–350.

[27] K.-C. Leung, V. O. Li, and D. Yang, "An overview of packet reordering in transmission control protocol (tcp): problems, solutions, and challenges," *IEEE transactions on parallel and distributed systems*, vol. 18, no. 4, pp. 522–535, 2007.

[28] Y. Liu, J. Cao, C. Liu, K. Ding, and L. Jin, "Datasets for large language models: A comprehensive survey," 2024. [Online]. Available: https://arxiv.org/abs/2402.18041

[29] Loadbalancer, "Enterprise max." [Online]. Available: https://www.loadbalancer.org/products/hardware/enterprise-max/

[30] J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, p. 39–53, apr 2004. [Online]. Available: https://doi.org/10.1145/997150.997156

[31] T. Mori, R. Kawahara, S. Naito, and S. Goto, "On the characteristics of internet traffic variability: spikes and elephants," in *2004 International Symposium on Applications and the Internet. Proceedings.*, 2004, pp. 99–106.

[32] T. Mori, M. Uchida, R. Kawahara, J. Pan, and S. Goto, "Identifying elephant flows through periodically sampled packets," in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 115–120. [Online]. Available: https://doi.org/10.1145/1028788.1028803

[33] P. A. Networks, "Intelligent traffic offload service for vm-series on kvm." [Online]. Available: https://docs.paloaltonetworks.com/pan-os/10-1/pan-os-new-features/virtualization-features/intelligent-traffic-offload

[34] P. A. Networks, "Pa-series next-generation firewalls - hardware architectures." [Online]. Available: https://www.paloaltonetworks.com/resources/pa-series-next-generation-firewalls-hardware-architectures

[35] P. A. Networks, "Session distribution policies." [Online]. Available: https://docs.paloaltonetworks.com/pan-os/10-1/pan-os-networking-admin/session-settings-and-timeouts/session-distribution-policies-overview

[36] P. A. Networks, *Strata by Palo Alto Networks — PA-7000 Series — Datasheet PA-7000 Series Palo Alto Networks PA-7000 Series ML-Powered*. [Online]. Available: https://www.paloaltonetworks.com/apps/pan/public/downloadResource?pagePath=/content/pan/en_US/resources/datasheets/pa-7000-series

[37] P. A. Networks, *Strata by Palo Alto Networks — PA-7500 — Datasheet*. [Online]. Available: https://www.paloaltonetworks.com/apps/pan/public/downloadResource?pagePath=/content/pan/en_US/resources/datasheets/pa-7500

[38] K. Neupane, R. Haddad, and L. Chen, "Next generation firewall for network security: A survey," in *SoutheastCon 2018*, 2018, pp. 1–6.

[39] OMNeT++, "Omnet++ discrete event simulator," 2019. [Online]. Available: https://omnetpp.org/

[40] Opnsense, "Opnsense, a true open source security platform and more - opnsense® is a true open source firewall and more." [Online]. Available: https://opnsense.org/

[41] S. Perreault, I. Yamagata, S. Miyakawa, A. Nakagawa, and H. Ashida, "Common requirements for carrier-grade nats (cgns)," Tech. Rep., 2013.

[42] T. Radivilova, L. Kirichenko, D. Ageyev, M. Tawalbeh, and V. Bulakh, "Decrypting ssl/tls traffic for hidden threats detection," in *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*. IEEE, 2018, pp. 143–146.
[43] M. Rasiukevicius, "rmind/npf," Mar 2024. [Online]. Available: https://github.com/rmind/npf
[44] Samtec, "Examax® high-speed backplane system — samtec." [Online]. Available: https://www.samtec.com/high-speed-board-to-board/backplane/examax/
[45] R. Shashidhara, T. Stamler, A. Kaufmann, and S. Peter, "Flextoe: Flexible tcp offload with fine-grained parallelism," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. Renton, WA: USENIX Association, Apr. 2022. [Online]. Available: https://www.usenix.org/conference/nsdi22/presentation/shashidhara
[46] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, "Blindbox: Deep packet inspection over encrypted traffic," in *Proceedings of the 2015 ACM conference on special interest group on data communication*, 2015, pp. 213–226.
[47] B. Soewito and C. E. Andhika, "Next generation firewall for improving security in company and iot network," in *2019 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, 2019, pp. 205–209.
[48] Suricata, "Home." [Online]. Available: https://suricata.io/
[49] T. Vermeiren, E. Borghs, and B. Haaodorens, "Evaluation of software techniques for parallel packet processing on multi-core processors," in *First IEEE Consumer Communications and Networking Conference, 2004. CCNC 2004.* IEEE, 2004, pp. 645–647.
[50] X. Wang, Y. Hong, H. Chang, K. Park, G. Langdale, J. Hu, and H. Zhu, "Hyperscan: A fast multi-pattern regex matcher for modern cpus," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*. Boston, MA: USENIX Association, Feb. 2019, pp. 631–648. [Online]. Available: https://www.usenix.org/conference/nsdi19/presentation/wang-xiang
[51] X.-L. Wu, W.-M. Li, F. Liu, and H. Yu, "Packet size distribution of typical internet applications," in *2012 International Conference on Wavelet Active Media Technology and Information Processing (ICWAMTIP)*, 2012, pp. 276–281.
[52] H. Xu and B. Li, "Tinyflow: Breaking elephants down into mice in data center networks," in *2014 IEEE 20th International Workshop on Local Metropolitan Area Networks (LANMAN)*, 2014, pp. 1–6.
[53] Z. Yang, Y. Cui, B. Li, Y. Liu, and Y. Xu, "Software-defined wide area network (sd-wan): Architecture, advances and opportunities," in *2019 28th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2019, pp. 1–9.
[54] Zenarmor, "Zenarmor - agile service edge security." [Online]. Available: https://www.zenarmor.com/
[55] Z. Zhao, H. Sadok, N. Atre, J. C. Hoe, V. Sekar, and J. Sherry, "Achieving 100gbps intrusion prevention on a single server," in *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. USENIX Association, Nov. 2020, pp. 1083–1100. [Online]. Available: https://www.usenix.org/conference/osdi20/presentation/zhao-zhipeng
[56] W. Zhu, P. Li, B. Luo, H. Xu, and Y. Zhang, "Research and implementation of high performance traffic processing based on intel dpdk," in *2018 9th international symposium on parallel architectures, algorithms and programming (PAAP)*. IEEE, 2018, pp. 62–68.
[57] Zybervr, "how to choose a pc vr link cable for your vr headset_2022," Dec 2022. [Online]. Available: https://zybervr.com/blogs/news/how-to-choose-a-pc-vr-link-cable-for-your-vr-headset