Technology Guide

Optimize NGFW Performance with Intel® Xeon® Processors on Public Cloud

Authors

Xiang Wang

Jayprakash Patidar

Declan Doherty

Eric Jones

Subhiksha Ravisundar

Heqing Zhu

1 Introduction

Next-generation firewalls (NGFWs) are at the core of network security solutions. Traditional firewalls perform stateful traffic inspection, typically based on port and protocol which cannot effectively defend against modern malicious traffic. NGFWs evolve and expand upon traditional firewalls with advanced deep packet inspection capabilities, including intrusion detection/prevention systems (IDS/IPS), malware detection, application identification and control, etc.

NGFWs are compute-intensive workloads performing, for example, cryptographic operations for network traffic encryption and decryption and heavy rule matching for detecting malicious activities. Intel delivers core technologies to optimize NGFW solutions. Intel processors are equipped with various instruction set architectures (ISAs), including Intel® Advanced Encryption Standard New Instructions (Intel® AES-NI) and Intel® QuickAssist Technology (Intel® QAT) which significantly accelerate crypto performance. Intel also invests in software optimizations including those for Hyperscan. Hyperscan is a high-performance string and regular expression (regex) matching library. It leverages single instruction multiple data (SIMD) technology on Intel processors to boost pattern-matching performance. Hyperscan integration into NGFW IPS systems such as Snort can improve performance by up to 3x on Intel processors.

NGFWs are often delivered as a security appliance deployed in the demilitarized zone (DMZ) of enterprise data centers. However, there is a strong demand for NGFW virtual appliances or software packages that can be deployed to the public cloud, in enterprise data centers, or at network edge locations. This software deployment model frees up enterprise IT from the operations and maintenance overhead associated with physical appliances. It improves system scalability and provides flexible procurement and purchasing options.

An increasing number of enterprises are embracing public cloud deployments of NGFW solutions. A key reason for this is the cost advantage of running virtual appliances in the cloud. Yet, since CSPs offer a multitude of instance types with varying compute characteristics and pricing, selecting the instance with the best TCO for NGFW can be challenging.

This paper introduces an NGFW reference implementation from Intel, optimized with Intel technologies, including Hyperscan. It offers a reliable proof-point for NGFW performance characterization on Intel platforms. It is included as part of Intel's NetSec Reference Software package. We also provide the Multi-Cloud Networking Automation Tool (MCNAT) in the same package to automate the deployment of the NGFW reference implementation on select public cloud providers. MCNAT simplifies TCO analysis for different compute instances and guides users to the optimal compute instance for NGFW.

Please contact authors to learn more about the NetSec Reference Software package.

Table of Contents

Figures

Tables

Document Revision History

Revision	Date	Description
001	March 2025	Initial release.

1.1 Terminology

Table 1. Terminology

Abbreviation	Description
DFA	Deterministic Finite Automaton
DPI	Deep Packet Inspection
HTTP	Hypertext Transfer Protocol
IDS/IPS	Intrusion Detection and Prevention System
ISA	Instruction Set Architecture
MCNAT	Multi-Cloud Networking Automation Tool
NFA	Non-deterministic Finite Automaton
NGFW	Next-generation Firewall
PCAP	Packet Capture
PCRE	Perl Compatible Regular Expressions Library
Regex	Regular Expression
SASE	Secure Access Service Edge
SIMD	Single Instruction Multiple Data Technology
TCP	Transmission Control Protocol
URI	Uniform Resource Identifier
WAF	Web Application Firewall

1.2 Reference Documentation

Table 2. Reference Documents

Reference	Source
Intel® Xeon® Scalable Platform Built for Most Sensitive Workloads	https://www.intc.com/news-events/press-releases/detail/1423/intel-xeon-scalable-platform-built-for-most-sensitive
Snort	https://www.snort.org/
Snort Talos Rules	https://www.snort.org/downloads#rules
Hyperscan	https://www.intel.com/content/www/us/en/developer/articles/technical/introduction-to-hyperscan.html
Hyperscan and Snort Integration	https://www.intel.com/content/www/us/en/developer/articles/technical/hyperscan-and-snort-integration.html
Hyperscan: A Fast Multi-Pattern Regex Matcher for Modern CPUs	https://www.usenix.org/conference/nsdi19/presentation/wang-xiang
Teddy: An Efficient SIMD-based Literal Matching Engine for Scalable Deep Packet Inspection	https://dl.acm.org/doi/10.1145/3472456.3473512
Intel® 64 and IA-32 Architectures Software Developer Manuals	https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html
Intel® Intrinsics Guide	https://www.intel.com/content/www/us/en/docs/intrinsics-guide/index.html
Accelerating Suricata Throughput Performance Using Hyperscan Pattern-Matching Software	https://www.intel.com/content/dam/www/public/us/en/documents/solution-briefs/hyperscan-scalability-solution-brief.pdf
Suricata	https://suricata.io/
Hyperscan in Suricata: State of the Union	https://suricon.net/wp-content/uploads/2016/11/SuriCon2016_GeoffLangdale.pdf
Accelerate Snort Performance with Hyperscan and Intel® Xeon® Processors on Public Clouds	https://networkbuilders.intel.com/solutionslibrary/accelerate-snort-performance-with-hyperscan-and-intel-xeon-processors-on-public-clouds
Next Generation Firewall – Optimizations with 4th Gen Intel® Xeon® Scalable Processor	https://networkbuilders.intel.com/solutionslibrary/next-generation-firewall-optimizations-solution-brief
Optimize Throughput and Power Efficiency for Next-Generation Firewalls	https://www.intel.com/content/www/us/en/products/docs/processors/xeon-accelerated/network/xeon6-firewall-solution-brief.html
NetSec Software Package	https://www.intel.com/content/www/us/en/secure/design/confidential/software-kits/kit-details.html?kitId=853965

2 Background and Motivation

Today, most NGFW vendors have extended their footprints from physical NGFW appliances to virtual NGFW solutions that can be deployed in the public cloud. Public cloud NGFW deployments are seeing increased adoption due to the following benefits:

- **Scalability:** easily scale up or scale down cross-geo compute resources to meet performance requirements.
- **Cost effectiveness:** flexible subscription to allow pay per use. Eliminates capital expenditure (capex) and reduces operational costs associated with physical appliances.
- **Native integration with cloud services:** seamless integration with public cloud services such as networking, access controls and AI/ML tools.
- **Cloud workloads protection:** local traffic filtering for enterprise workloads hosted on public cloud.

The reduced cost of running the NGFW workload in the public cloud is an attractive proposition for enterprise use cases. However, selecting the instance with the best performance and TCO for NGFW is challenging, given a wide range of cloud instance options are available with various CPUs, memory sizes, IO bandwidth, and each is priced differently. We have developed NGFW Reference Implementation to help with performance and TCO analysis of different public cloud instances based on Intel processors. We will demonstrate performance and performance per dollar metrics as a guide for choosing the right Intel-based instances for NGFW solutions on public cloud services such as AWS and GCP.

3 NGFW Reference Implementation

Intel developed the NetSec Reference Software package (latest release 25.05) which delivers optimized reference solutions leveraging ISAs and accelerators available in the newest Intel CPUs and platforms to demonstrate optimized performance at the on-prem enterprise infrastructure and on the cloud. The reference software is available under Intel Proprietary License (IPL). The key highlights of this software package are:

- Includes a broad portfolio of reference solutions for networking and security, AI frameworks for cloud and enterprise data centers and edge locations.
- Allows time to market and rapid adoption of Intel technologies.
- Source code is available that allows replicating deployment scenarios and testing environments on Intel platforms.

Please contact authors to learn more about obtaining the latest release of the NetSec Reference Software.

As a critical part of NetSec Reference Software package, NGFW reference implementation drives the NGFW performance characteristics and TCO analysis on Intel platforms. We deliver seamless integration of Intel technologies such as Hyperscan in the NGFW reference implementation. It builds a solid foundation for NGFW analysis on Intel platforms. Since different Intel hardware platforms offer different capabilities from compute to IO, the NGFW reference implementation presents a clearer view of platform capabilities for NGFW workloads and helps show performance comparisons between generations of Intel processors. It delivers thorough insights on metrics, including compute performance, memory bandwidth, IO bandwidth, and power consumption. Based on performance test results, we can further conduct TCO analysis (with performance per dollar) on Intel platforms used for NGFW.

The latest release (25.05) of NGFW reference implementation includes the following key features:

- Basic stateful firewall
- Intrusion Prevention System (IPS)
- Support of cutting-edge Intel processors including Intel® Xeon® 6 processors, Intel Xeon 6 SoC, etc.

Future releases are planned to implement the following additional features:

- VPN inspection: IPsec decryption of traffic for content inspection
- TLS inspection: a TLS Proxy to terminate the connections between a client and a server and then perform content inspection on the plaintext traffic.

3.1 System Architecture

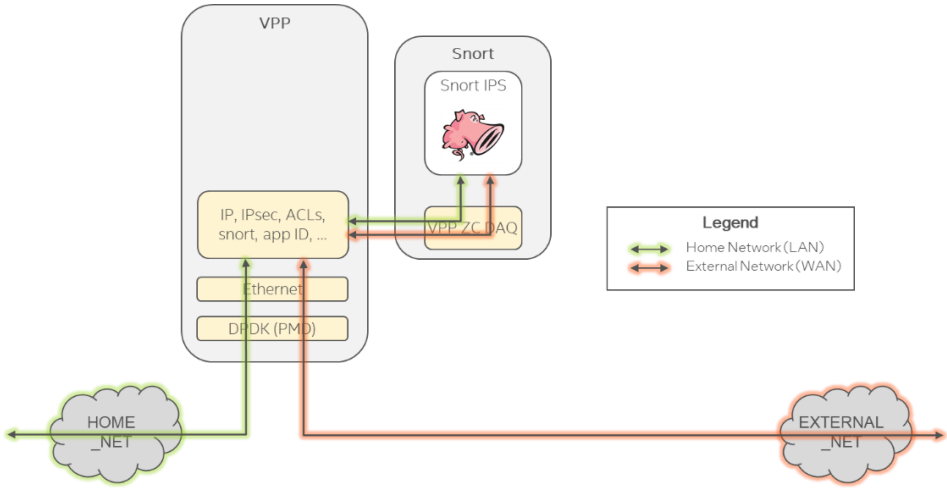

Figure 1. NGFW Reference Architecture

Figure 1 shows the overall system architecture. We leverage open-source software as the foundation to build the system:

- VPP provides a high-performance data plane solution with basic stateful firewall functions, including stateful ACLs. We spawn multiple VPP threads with configured core affinity. Each VPP worker thread is pinned to a dedicated CPU core or an execution thread.
- Snort 3 is chosen as IPS, which supports multi-threading. Snort worker threads are pinned to dedicated CPU cores or execution threads.
- Snort and VPP are integrated using the Snort plugin to VPP. This uses a set of queue pairs for sending packets between VPP and Snort. The queue pairs and the packets themselves are stored in shared memory. We developed a new Data Acquisition (DAQ) component for Snort, which we call the VPP Zero Copy (ZC) DAQ. This implements the Snort DAQ API functions to receive and transmit packets by reading from and writing to the relevant queues. Because the payload is in shared memory, we consider this a Zero-Copy implementation.

Since Snort 3 is a compute-intensive workload that requires more computing resources than data plane processing, we are trying to configure an optimized processor core allocation and balance between the number of VPP threads and Snort3 threads to get the highest system level performance on the running hardware platform.

Figure 2 (on page 6) shows the graph node within VPP, including those that are part of the ACL and Snort plugins. We developed two new VPP graph nodes:

- **snort-enq**: makes a load-balancing decision about which Snort thread should process the packet and then enqueues the packet to the corresponding queue.
- **snort-deq**: implemented as an input node that polls from multiple queues, one per Snort worker thread.

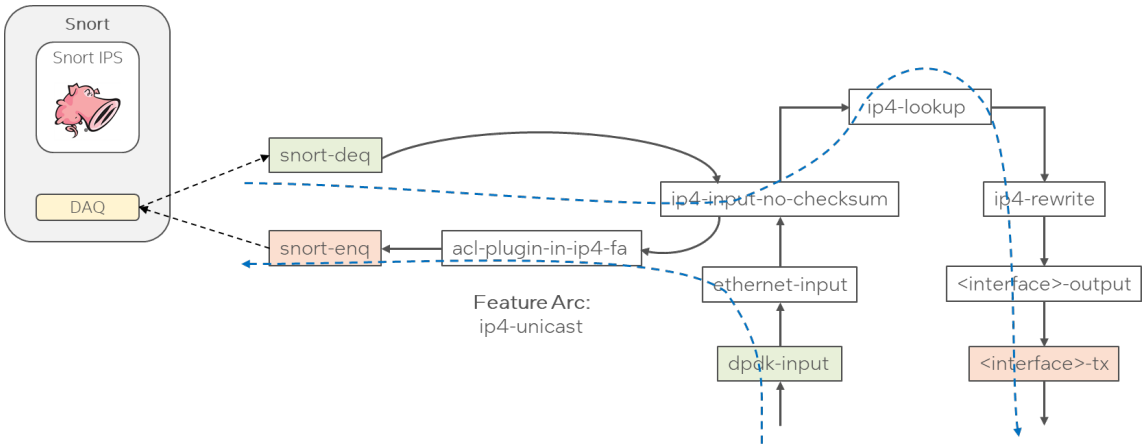

Figure 2. NGFW Reference Architecture with VPP Graph Nodes

3.2 Intel Optimizations

Our NGFW reference implementation takes advantage of the following optimizations:

- Snort leverages the Hyperscan high-performance multiple regex matching library to provide a significant boost in performance compared to the default search engine in Snort. Figure 3 highlights Hyperscan integration with Snort to accelerate both literal machng and regex matching performance. Snort 3 provides native integration with Hyperscan where users can turn on Hyperscan either via config file or command line options.

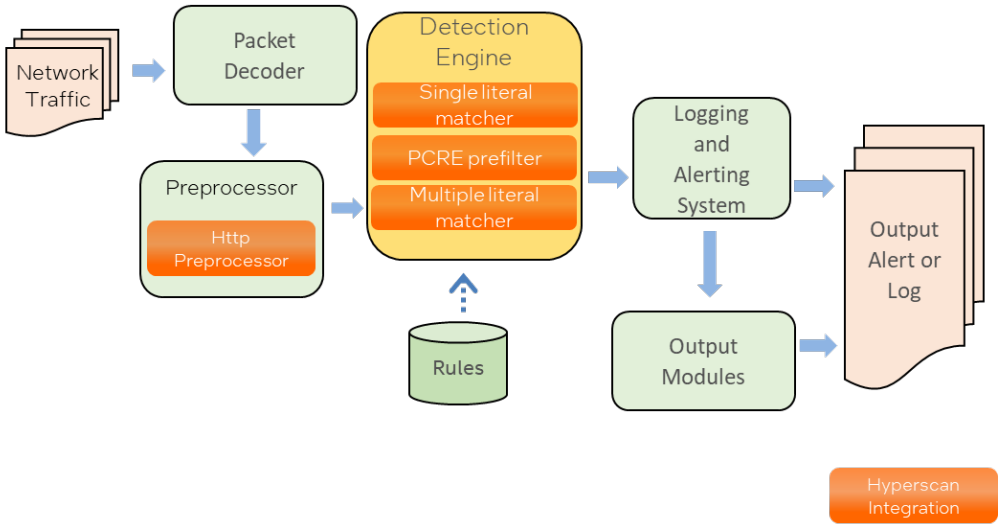

Figure 3. Snort with Hyperscan Integration

- VPP takes advantage of Receive Side Scaling (RSS) in Intel® Ethernet Network Adapters to distribute traffic across multiple VPP worker threads.
- Intel QAT and Intel AVX-512 instructions: Future releases that support IPsec and TLS will be taking advantage of crypto acceleration technologies from Intel. Intel QAT accelerates crypto performance, especially the public key cryptography which is widely used for establishing network connections. Intel AVX-512 also boosts cryptographic performance, including VPMADD52 (multiply and accumulation operations), vector AES (vector version of the Intel AES-NI instructions), vPCLMUL (vectorized carry-less multiply, used to optimize AES-GCM), and Intel® Secure Hash Algorithm - New Instructions (Intel® SHA-NI).

4 Cloud Deployment of NGFW Reference Implementation

4.1 System Configuration

Table 3. Test configurations

Metric	Value
Use Case	Cleartext Inspection (FW + IPS)
Traffic Profile	HTTP 64KB GET (1 GET per Connection)
VPP ACLs	Yes (2 stateful ACLs)
Snort Rules	Lightspd (~49k rules)
Snort Policy	Security (~21k rules enabled)

We focus on cleartext inspection scenarios based on use cases and KPIs in RFC9411. The traffic generator could create 64KB HTTP transactions with 1 GET request per connection. ACLs are configured to allow IPs in the specified subnets. We adopted Snort Lightspd ruleset and the security policy from Cisco for benchmarking. There was also a dedicated server to serve requests from traffic generators.

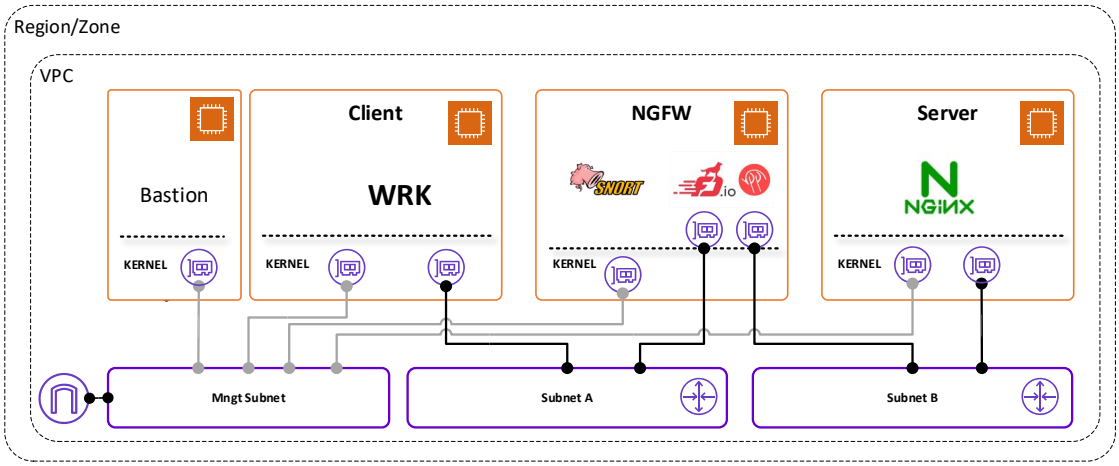

Figure 4. AWS System Topology

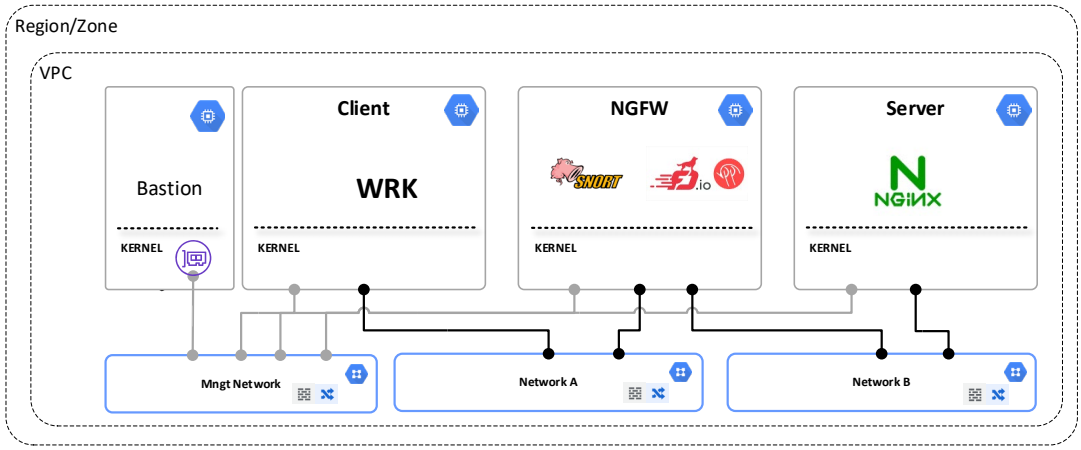

Figure 5. GCP System Topology

As shown in Figure 4 and Figure 5, the system topology includes three primary instance nodes: a client, a server and a proxy for public cloud deployment. There is also a bastion node to serve connections from user. Both client (running WRK) and server (running Nginx) have a single dedicated data-plane network interface, and the proxy (running NGFW) has two data-plane network interfaces for testing. Data-plane network interfaces are attached to dedicated subnet A (client-proxy) and subnet B (proxy-server) which maintain isolation from instance management traffic. Dedicated IP address ranges are defined with corresponding routing and ACL rules programmed onto the infrastructure to allow flow of traffic.

4.2 System Deployment

MCNAT is a software tool developed by Intel that provides automation for seamless networking workload deployments on public cloud and offers suggestions on selecting the best cloud instance based on performance and cost.

MCNAT is configured through a series of profiles, each defining the variables and settings required for each instance. Each instance type has its own profile which can then be passed to the MCNAT CLI tool to deploy that specific instance type on a given cloud service provider (CSP). Example command line usage is shown below and in Table 4.

```
# ./mcnat.py --deploy -u user -c aws.oregon -s ngfw-intel -p c7i-xlarge
```

Table 4. MCNAT Command Line Usage

Option	Description
--deploy	Instructs the tool to create a new deployment
-u	Defines which user credentials to use
-c	CSP to create deployment on (AWS, GCP, etc)
-s	Scenario to deploy
-p	Profile to use

The MCNAT command line tool can build and deploy instances in a single step. Once the instance is deployed, the post configuration steps create the necessary SSH configuration to allow the instance to be accessed.

4.3 System Benchmarking

Once MCNAT has deployed the instances, all performance tests can run using the MCNAT application toolkit.

First, we need to configure test cases at tools/mcn/applications/configurations/ngfw-intel/ngfw-intel.json as below:

```
"description": "Throughput test, 64KiB, Hyperscan",
"proxy": {
    "ngfw-intel": {
        "test_tag": "Throughput test",
        "vpp_cores": 1,
        "snort_cores": 1,
        "vpp_affinity" : "0,2",
        "snort_affinity" : "0,1,3",
        "hyperscan": true
    }
}
```

Then we can use the example command below to launch the test. The DEPLOYMENT_PATH is where the target environment deployment state is stored, e.g., tools/mcn/infrastructure/infrastructure/examples/ngfw-intel/gcp/terraform.tfstate.d/tfws_default.

```
# python -m mcn_application_toolkit run --deployment-path DEPLOYMENT_PATH -f
configurations/ngfw-intel/ngfw-intel.json ngfw-intel
```

It runs NGFW with a given set of rules on http traffic generated by WRK on client, while pinning a range of CPU cores, to gather a full set of performance numbers for the instance under test. When the tests are completed, all the data is formatted as a csv and returned to the user.

5 Performance and Cost Evaluation

In this section, we compare NGFW deployments on different cloud instances based on Intel Xeon processors at AWS and GCP. This gives guidance on finding the most suitable cloud instance type for NGFW based on performance and cost. We choose instances with 4 vCPUs as they are recommended by most NGFW vendors. Results on AWS and GCP include:

- NGFW performance on small instance types that host 4 vCPUs with Intel® Hyper-Threading Technology (Intel® HT Technology) and Hyperscan enabled.
- Generation-to-generation **performance** gains from 1st Gen Intel Xeon Scalable processors to 5th Gen Intel Xeon Scalable processors.
- Generation-to-generation **performance per dollar** gain from 1st Gen Inte® Xeon Scalable processors to 5th Gen Intel Xeon Scalable processors.

5.1 AWS Deployment

5.1.1 Instance Type List

Table 5. AWS Instances and On-demand Hour Rates

Instance Type	CPU Model	vCPU	Memory (GB)	Network performance (Gbps)	On-demand hourly rate ($)
c5-xlarge	2nd Gen Intel® Xeon® Scalable processors	4	8	10	0.17
c5n-xlarge	1st Gen Intel® Xeon® Scalable processors	4	10.5	25	0.216
c6i-xlarge	3rd Gen Intel® Xeon® Scalable processors	4	8	12.5	0.17
c6in-xlarge	3rd Gen Intel Xeon Scalable processors	4	8	30	0.2268
c7i-xlarge	4th Gen Intel® Xeon® Scalable processors	4	8	12.5	0.1785

Table 5 shows the overview of AWS instances we use. Please refer to Platform Configuration for more platform details. It also lists the on-demand hourly rate (https://aws.amazon.com/ec2/pricing/on-demand/) for all instances. The above was the on-demand rate at the time of publishing this paper and focuses on the US west coast. The on-demand hourly rate might vary with the region, availability, corporate accounts, and other factors.

5.1.2 Results

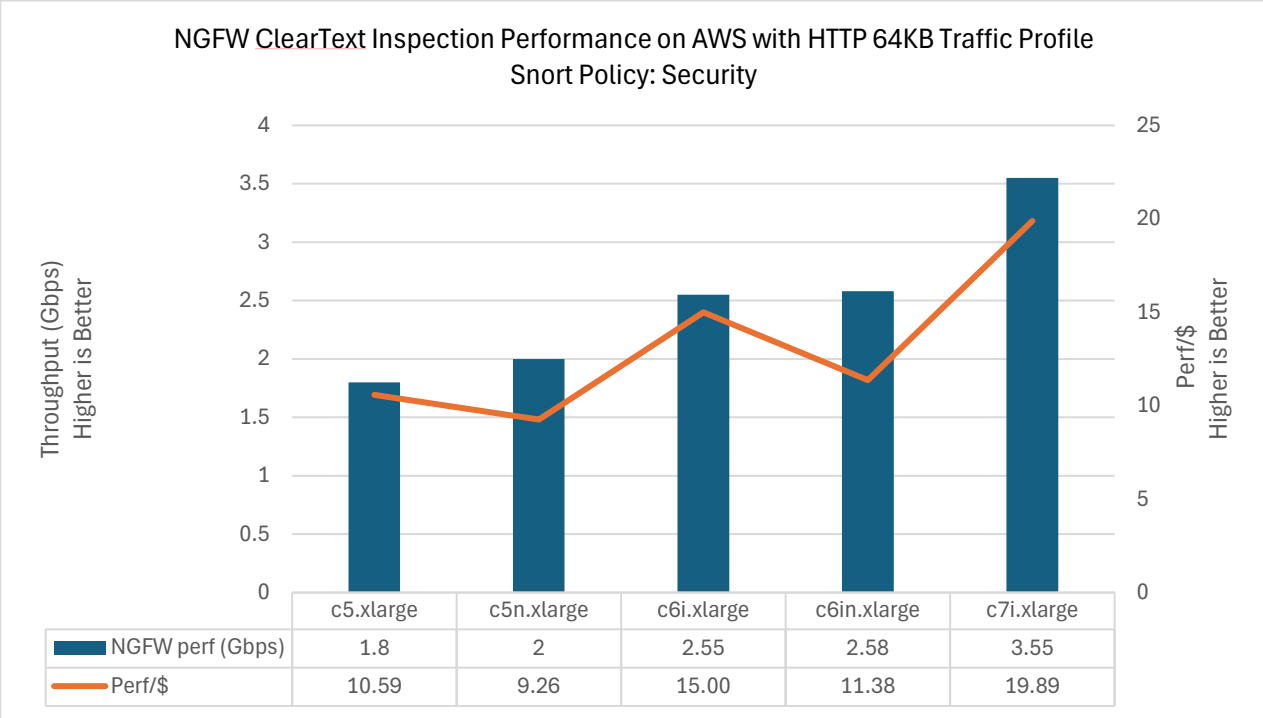

Figure 6. NGFW Reference Software Performance and Performance per Dollar on AWS

Figure 6 compares performance and performance per hour rate on all the instance types mentioned thus far:

- Performance improved with instances based on newer generations of Intel Xeon processors. Upgrading from c5.xlarge (based on 2nd Gen Intel Xeon Scalable processor) to c7i.xlarge (based on 4th Gen Intel Xeon Scalable processor) shows a **1.97x performance improvement**.
- Performance per dollar improved with instances based on newer generations of Intel Xeon processors. Upgrading from c5n.xlarge (based on 1st Gen Intel Xeon Scalable processor) to c7i.xlarge (based on 4th Gen Intel Xeon Scalable processor) shows a **1.88x performance/hour rate improvement**.

5.2 GCP Deployment

5.2.1 Instance Type List

Table 6. GCP Instances and On-demand Hour Rates

Instance Type	CPU Model	vCPU	Memory (GB)	Default egress bandwidth (Gbps)	On-demand hourly rate ($)
n1-std-4	1st Gen Intel® Xeon® Scalable processors	4	15	10	0.189999
n2-std-4	3rd Gen Intel® Xeon® Scalable processors	4	16	10	0.194236
c3-std-4	4th Gen Intel® Xeon® Scalable processors	4	16	23	0.201608
n4-std-4	5th Gen Intel® Xeon® Scalable processors	4	16	10	0.189544
c4-std-4	5th Gen Intel® Xeon® Scalable processors	4	15	23	0.23761913

Table 6 shows the overview of GCP instances we use. Please refer to Platform Configuration for more platform details. It also lists the on-demand hourly rate (https://cloud.google.com/compute/vm-instance-pricing?hl=en) for all instances. The above was the on-demand rate at the time of publishing this paper and focuses on the US west coast. The on-demand hourly rate might vary with the region, availability, corporate accounts, and other factors.

5.2.2 Results

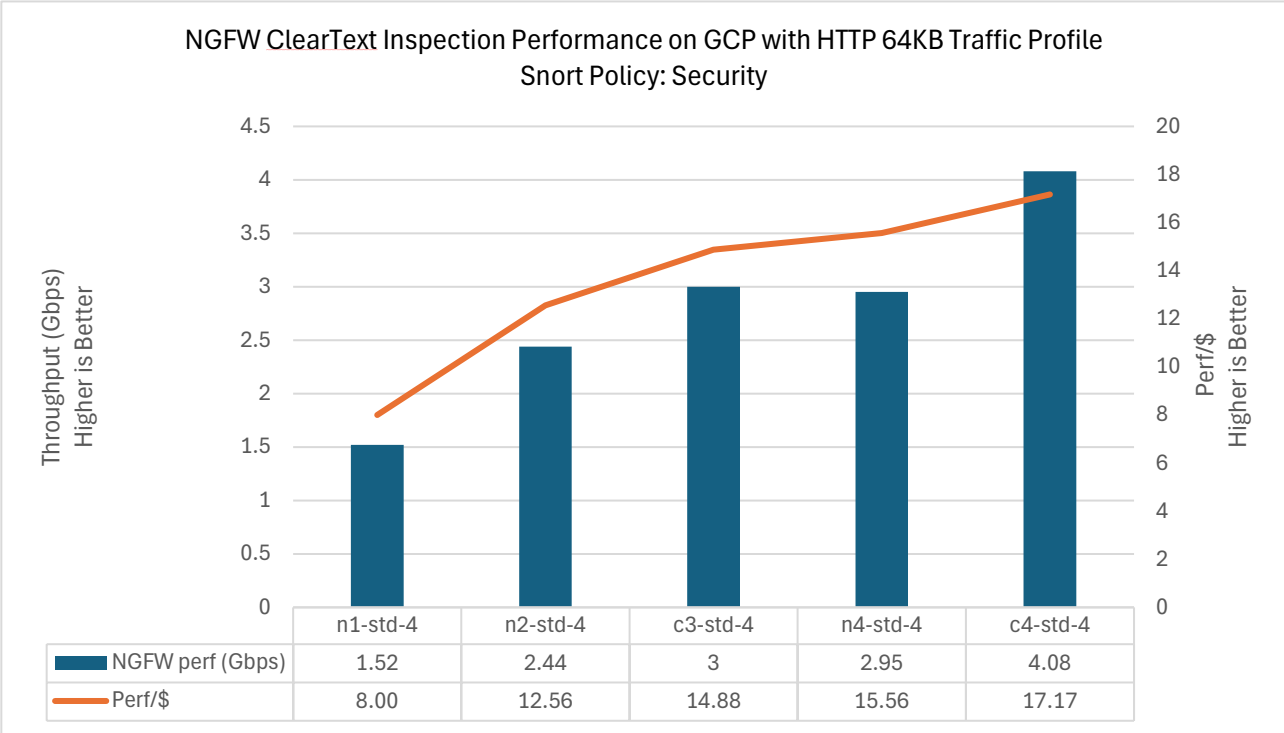

Figure 7. NGFW Reference Software Performance and Performance per Dollar on GCP

Figure 7 compares performance and performance per hour rate on all the instance types mentioned thus far:

- Performance improved with instances based on newer generations of Intel Xeon processors. Upgrading from n1-std-4 (based on 1st Gen Intel Xeon Scalable processor) to c4-std-4 (based on 5th Gen Intel Xeon Scalable processor) shows a **2.68x performance improvement.**
- Performance per dollar improved with instances based on newer generations of Intel Xeon processors. Upgrading from n1-std-4 (based on 1st Gen Intel Xeon Scalable processor) to c4-std-4 (based on 5th Gen Intel Xeon Scalable processor) shows a **2.15x performance/hour rate improvement**.

6 Summary

With the increasing adoption of multi- and hybrid-cloud deployment models, delivering NGFW solutions on public cloud provides consistent protection across environments, scalability to meet security requirements, and simplicity with minimal maintenance efforts. Network security vendors offer NGFW solutions with a variety of cloud instance types on public cloud. It's critical to minimize total cost of ownership (TCO) and maximize return on investment (ROI) with the right cloud instance. The key factors to consider include compute resources, network bandwidth, and price. We used NGFW reference implementation as the representative workload and leveraged MCNAT to automate the deployment and testing on different public cloud instance types. Based on our benchmarking, instances with the latest generation of Intel Xeon Scalable processors on AWS (powered by 4th Intel Xeon Scalable processors) and GCP (powered by 5th Intel Xeon Scalable processors) deliver both performance and TCO improvements. They improve the performance by up to 2.68x and the performance per hour rate by up to 2.15x over prior generations. This evaluation generates solid references on selecting Intel based public cloud instances for NGFW.

Appendix A Platform Configuration

Platform Configurations
c5-xlarge - "Test by Intel as of 03/17/25. 1-node, 1x Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz, 2 cores, HT On, Turbo On, Total Memory 8GB (1x8GB DDR4 2933 MT/s [Unknown]), BIOS 1.0, microcode 0x5003801, 1x Elastic Network Adapter (ENA), 1x 32G Amazon Elastic Block Store, Ubuntu 22.04.5 LTS, 6.8.0-1024-aws, gcc 11.4, NGFW 24.12, Hyperscan 5.6.1"
c5n-xlarge - "Test by Intel as of 03/17/25. 1-node, 1x Intel(R) Xeon(R) Platinum 8124M CPU @ 3.00GHz, 2 cores, HT On, Turbo On, Total Memory 10.5GB (1x10.5GB DDR4 2933 MT/s [Unknown]), BIOS 1.0, microcode 0x2007006, 1x Elastic Network Adapter (ENA), 1x 32G Amazon Elastic Block Store, Ubuntu 22.04.5 LTS, 6.8.0-1024-aws, gcc 11.4, NGFW 24.12, Hyperscan 5.6.1"
c6i-xlarge - "Test by Intel as of 03/17/25. 1-node, 1x Intel(R) Xeon(R) Platinum 8375C CPU @ 2.90GHz, 2 cores, HT On, Turbo On, Total Memory 8GB (1x8GB DDR4 3200 MT/s [Unknown]), BIOS 1.0, microcode 0xd0003f6, 1x Elastic Network Adapter (ENA), 1x 32G Amazon Elastic Block Store, Ubuntu 22.04.5 LTS, 6.8.0-1024-aws, gcc 11.4, NGFW 24.12, Hyperscan 5.6.1"
c6in-xlarge - "Test by Intel as of 03/17/25. 1-node, 1x Intel(R) Xeon(R) Platinum 8375C CPU @ 2.90GHz, 2 cores, HT On, Turbo On, Total Memory 8GB (1x8GB DDR4 3200 MT/s [Unknown]), BIOS 1.0, microcode 0xd0003f6, 1x Elastic Network Adapter (ENA), 1x 32G Amazon Elastic Block Store, Ubuntu 22.04.5 LTS, 6.8.0-1024-aws, gcc 11.4, NGFW 24.12, Hyperscan 5.6.1"
c7i-xlarge - "Test by Intel as of 03/17/25. 1-node, 1x Intel(R) Xeon(R) Platinum 8488C CPU @ 2.40GHz, 2 cores, HT On, Turbo On, Total Memory 8GB (1x8GB DDR4 4800 MT/s [Unknown]), BIOS 1.0, microcode 0x2b000620, 1x Elastic Network Adapter (ENA), 1x 32G Amazon Elastic Block Store, Ubuntu 22.04.5 LTS, 6.8.0-1024-aws, gcc 11.4, NGFW 24.12, Hyperscan 5.6.1"
n1-std-4 – "Test by Intel as of 03/17/25. 1-node, 1x Intel(R) Xeon(R) CPU @ 2.00GHz, 2 cores, HT On, Turbo On, Total Memory 15GB (1x15GB RAM []), BIOS Google, microcode 0xffffffff, 1x device, 1x 32G PersistentDisk, Ubuntu 22.04.5 LTS, 6.8.0-1025-gcp, gcc 11.4, NGFW 24.12, Hyperscan 5.6.1"
n2-std-4 - Test by Intel as of 03/17/25. 1-node, 1x Intel(R) Xeon(R) CPU @ 2.60GHz, 2 cores, HT On, Turbo On, Total Memory 16GB (1x16GB RAM []), BIOS Google, microcode 0xffffffff, 1x device, 1x 32G PersistentDisk, Ubuntu 22.04.5 LTS, 6.8.0-1025-gcp, gcc 11.4, NGFW 24.12, Hyperscan 5.6.1"
c3-std-4 - Test by Intel as of 03/14/25. 1-node, 1x Intel(R) Xeon(R) Platinum 8481C CPU @ 2.70GHz @ 2.60GHz, 2 cores, HT On, Turbo On, Total Memory 16GB (1x16GB RAM []), BIOS Google, microcode 0xffffffff, 1x Compute Engine Virtual Ethernet [gVNIC], 1x 32G nvme_card-pd, Ubuntu 22.04.5 LTS, 6.8.0-1025-gcp, gcc 11.4, NGFW 24.12, Hyperscan 5.6.1"
n4-std-4 - Test by Intel as of 03/18/25. 1-node, 1x Intel(R) Xeon(R) PLATINUM 8581C CPU @ 2.10GHz, 2 cores, HT On, Turbo On, Total Memory 16GB (1x16GB RAM []), BIOS Google, microcode 0xffffffff, 1x Compute Engine Virtual Ethernet [gVNIC], 1x 32G nvme_card-pd, Ubuntu 22.04.5 LTS, 6.8.0-1025-gcp, gcc 11.4, NGFW 24.12, Hyperscan 5.6.1"
c4-std-4 - Test by Intel as of 03/18/25. 1-node, 1x Intel(R) Xeon(R) PLATINUM 8581C CPU @ 2.30GHz, 2 cores, HT On, Turbo On, Total Memory 15GB (1x15GB RAM []), BIOS Google, microcode 0xffffffff, 1x Compute Engine Virtual Ethernet [gVNIC], 1x 32G nvme_card-pd, Ubuntu 22.04.5 LTS, 6.8.0-1025-gcp, gcc 11.4, NGFW 24.12, Hyperscan 5.6.1"

Appendix B Intel NGFW Reference Software Configuration

Software Configuration	Software Version
Host OS	Ubuntu 22.04 LTS
Kernel	6.8.0-1025
Compiler	GCC 11.4.0
WRK	74eb9437
WRK2	44a94c17
VPP	24.02
Snort	3.1.36.0
DAQ	3.0.9
LuaJIT	2.1.0-beta3
Libpcap	1.10.1
PCRE	8.45
ZLIB	1.2.11
Hyperscan	5.6.1
LZMA	5.2.5
NGINX	1.22.1
DPDK	23.11

0425/XW/MK/PDF 365150-001US