White Paper

Communications Service Providers
5G Multi-Access Edge Computing

Reference Architecture for Confidential Computing on SKT 5G MEC

Table of Contents

1. Introduction

The mobile network operator represents a paradigm shift and delivers increased revenues to service-oriented businesses in the managed 5G B2B services sector. Part of this shift includes cloud-native network function (CNF), a new approach to building complex networking solutions based on the principles of cloud-native computing and microservices. These CNFs, running as private clouds inside telecommunications premises, can benefit from the same technology applied to public clouds.

One such emerging technology is confidential computing which addresses concerns about the confidentiality and alterability of data and applications. Confidential computing addresses threats such as malicious system administrators or other insiders at the cloud service provider (CSP), attackers exploiting vulnerabilities in the underlying cloud fabric, or other third parties accessing data without the data owner's consent.

This paper examines the performance impact of confidential computing on the SKT 5G MEC using Intel® SGX technology, a feature available on SGX Enclave Page Cache (EPC) models of 3rd generation Intel® Xeon® Scalable processors.

Key Object of Test

This document describes the results of testing the SKT 5G MEC software running on Intel® architecture server boards, including benchmarking data and instructions on how to replicate the tests. Telecom equipment manufacturers (TEMs) and independent software vendors (ISVs) can use the implementation guidelines from this work to optimize and further develop confidential computing solutions for high-performance production services.

The test objectives were:

- Compare the performance of an SGX-enabled encrypted Redis container with the original Redis container.
- Show use cases for privacy-preserving machine learning (PPML) to protect privacy-sensitive user data, and a trained model that uses the SGX-enabled Analytic Zoo E2E platform with ML/DL analytic frameworks.
- Validate interoperability between SGX and multiple application containers on SKT 5G MEC.

1.1 Confidential Computing Overview

Confidential computing integrates a stack of hardware and software in a way that mitigates some of this risk. At the base of that stack is the trusted execution environment (TEE), also called an enclave. It is where data and code are isolated and shielded from other software, including the operating system and the cloud service stack. The hardware ensures that code and data cannot be viewed or modified from outside of the TEE. Even with privileged root access, an authorization code is required to access data or run code in the TEE.

This protection is accomplished through a combination of physically encrypting a portion of memory and changing the memory access control so that previously privileged software (OS, hypervisor, etc.) can no longer access the data or application code within it. Developers can use libraries and extensions such as Intel® Software Guard Extensions (Intel® SGX) to create applications which use these enclaves.

The mobile network operator can provide developers an existing container application (new or existing) and run it securely on MEC through SGX-supported confidential nodes. Support, such as commercial confidential computing services, confidential containers, and virtual machines, can protect data integrity and confidentiality with hardware-based assurances in these use cases:

- block chain, secret storage, ML-inferencing IoT, and data integrity
- privacy-preserving machine learning and federated learning

1.2 Overview of SKT 5G MEC 2.0

Mobile edge computing (MEC) is a network architecture concept that enables cloud computing capabilities and an IT service environment at the edge of the mobile telecommunication network. MEC allows applications to run and perform related processing tasks closer to the mobile end user. By doing so, MEC can serve as a platform that provides services requiring low latency among the representative use cases of 5G, such as AR/VR, cloud-based gaming, and smart transportation. MEC platform is designed to be implemented at cellular base stations or other mobile edge locations and enables agile deployment of new applications and services for customers.

SK Telecom first developed the MEC platform in 2019 based on ETSI-compliant architecture. Since then, the MEC platform continues to evolve with the goal of being customer friendly, cloud-native, and easy to deploy. The name of the current MEC Platform is SKT MEC 2.0, composed as shown in Figure 1. below.

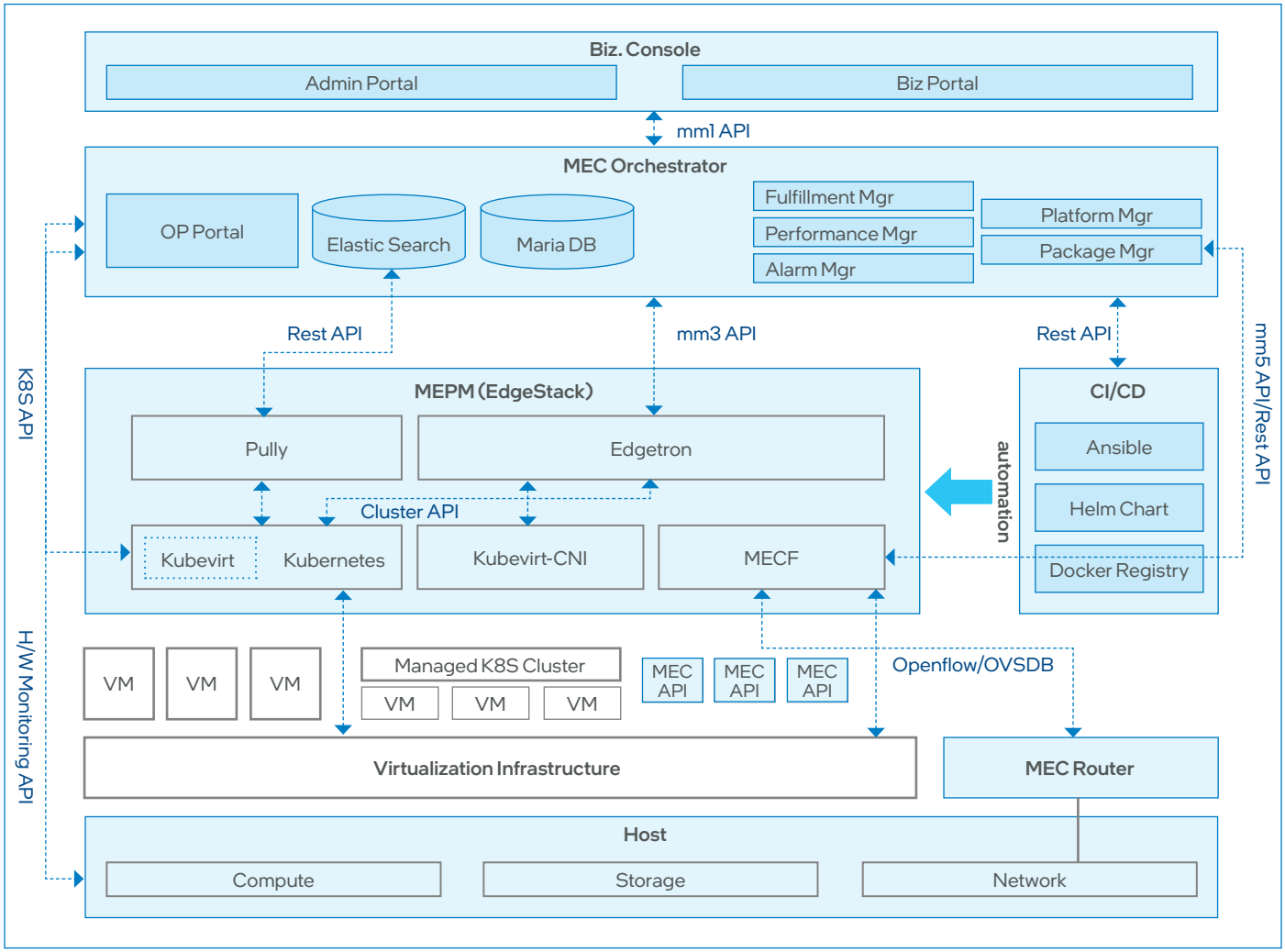

Figure 1. SKT 5G MEC 2.0 architecture

SKT MEC platform provides 1) MEC infrastructure function that provides a cloud computing environment; 2) the function to connect and manage traffic between the 5G network and the MEC infrastructure; 3) the function to provide a MEC API; 4) the function to manage infrastructure, resources, applications, and services; 5) customer-facing portal; and 6) A function that discovers the MEC site closest to the end user (see **Table 1**.).

Table 1: SKT 5G MEC 2.0 Functionality

Component	Main functionalities
1) MEPM (Edgestack)	• VM, Container, Managed K8S life cycle management • High-performance, security-enhanced cloud networking • VMware, public CSP infrastructure interworking
2) MEC Router	• Multi-access (LTE, 5G, WiFi) traffic control and management • GTP traffic encapsulation/decapsulation • Traffic offloading rule, load balancing, NAT, etc. • Providing per-application statistics, billing, and QoS information
3) MEC-as-a-Service	• SKT-specific MEC API: bearer-as-a-service, radio network information, location, security, etc. • Cloud-native IaaS/PaaS/SaaS: volume storage, load balancer, application monitoring, etc.
4) MEC Orchestrator	• Multi MEC site operation, management, and monitoring (FCAPS) • Public CSP resource management • Operational portal
5) Biz. Console	• MEC IaaS/PaaS/SaaS product offering • Customer-facing service portal and marketplace • Global MEC federation
6) MEC Director	• A function that discovers the MEC site closest to the end user • Management of additional services to distinguish MEC users

1.3 Key Technology

1.3.1 Intel Software Guard Extension

Intel Software Guard Extensions (Intel SGX) is a set of instructions that increases the security of application code and data. Developers can partition security-sensitive code and data into an "SGX Enclave" that is executed in a CPU-protected region.

Intel SGX offers the following enclave protections from known hardware and software attacks:

- OS or hypervisor, Intel management engine (ME), BIOS,firmware, drivers, System management module (Ring 2) can not read, access enclave.
- Enclave memory cannot be read or written from outside the enclave regardless of the current privilege level and CPU mode.
- Production enclaves cannot be debugged by software or hardware debuggers.
- The enclave environment cannot be entered through classic function calls, jumps, register manipulation, or stack manipulation. The only way to call an enclave function is through a new instruction that performs several protection checks.
- Enclave memory is encrypted using industry-standard encryption algorithms with replay protection. Tapping the memory or connecting the DRAM modules to another system will yield only encrypted data.
- The memory encryption key randomly changes every power cycle. The key is stored within the CPU and is not accessible.
- Data isolated within enclaves can only be accessed by code that shares the enclave.

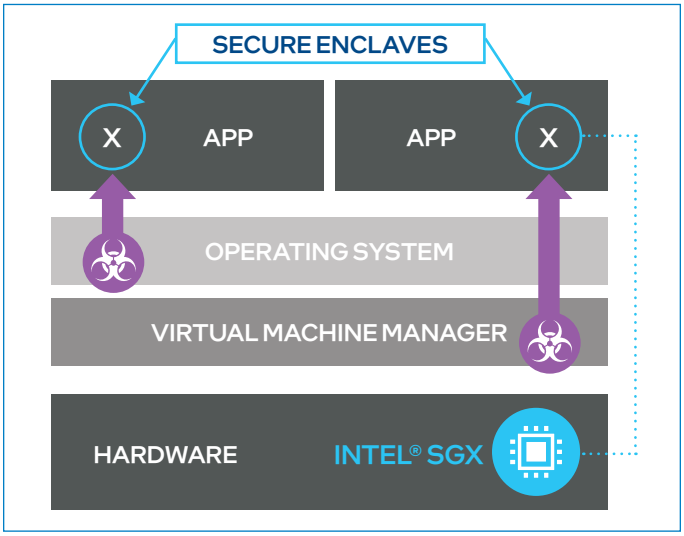

Figure 2. Intel SGX Security Model

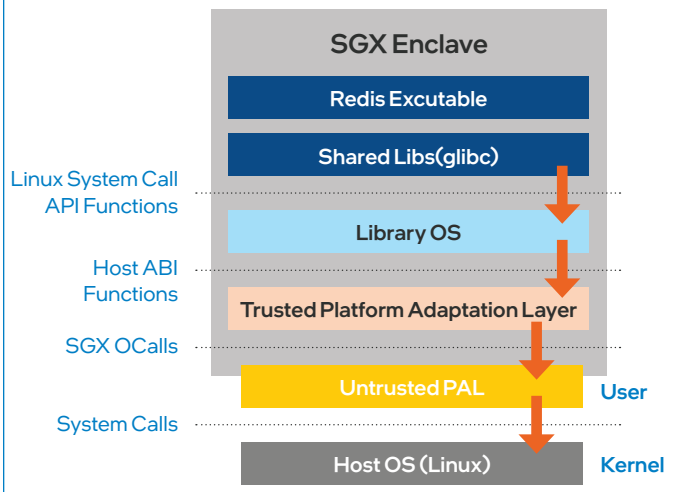

Figure 3. Intel SGX -Graphene Library OS

Figure 2 shows that the attack surface can be largely reduced after applying Intel SGX.

1.3.2 Graphene

Graphene is a lightweight library OS designed to run a single application with minimal host requirements. Figure 3 is a functional diagram of the Graphene library running with Intel SGX support (Graphene-SGX). Graphene can run applications in an isolated environment with benefits comparable to running a complete OS in a virtual machine, including guest customization, ease of porting to different operating systems, and process migration. Graphene supports native, unmodified Linux binaries on any platform. Currently, Graphene runs on Linux and Intel SGX enclaves on Linux platforms. In untrusted cloud and edge deployments, there is a strong desire to shield the whole application from the rest of the infrastructure. Graphene supports this "lift and shift" paradigm for bringing unmodified applications into Confidential Computing with Intel SGX. Graphene can protect applications from a malicious system stack with minimal porting effort. It is easy to port the Graphene solution to a new OS or platform.

Graphene-SGX turns an unmodified application into an enclave application with these features:

- Application-specific signature authenticates all binaries
- Syscalls are implemented inside enclaves
- Untrusted OS inputs are narrowed, redefined, and checked for validity

When enclaved applications communicate with untrusted services, they must perform OCalls as shown above in Figure 3. For example, network and file system OCalls must copy network packets and files to or from the enclave. This involves several sources of overhead that need to be resolved. So for that, Exitless(Switchless) feature that require additional CPU core on modern SGX-enabled Intel processors can reduce overhead challenge and, recommend to use Exitless only for single threaded applications like a redis. Latency can be prioritized over throughput by using core pinning with taskset, by isolating cores with isolcpus, or by disabling interrupts on cores via nohz_full.

1.3.3 Analytics Zoo/BigDL PPML

Analytics Zoo/BigDL, as shown in Figure 4 below, is a unified data analytics and AI platform for distributed TensorFlow, Keras, PyTorch, and Apache Spark/Flink and Ray. With Analytics Zoo/BigDL, the analytics frameworks (such as Spark, Flink, Ray), ML/DL frameworks (such as TensorFlow, PyTorch, OpenVINO etc.) and Python libraries (such as NumPy, Pandas, etc.) can run as integrated pieces in the LibOS (e.g, Graphene) in a protected manner. Analytics Zoo/BigDL also provides security features such as secure data access, secure gradient, and parameter management, which helps enable more privacy-preserving machine learning use cases such as federated learning.

With the Analytics Zoo/BigDL PPML platform, users can build a secure, end-to-end, distributed inference service pipeline, as shown in Figure 5. The inference service pipeline is constructed with Analytics Zoo/BigDL Cluster Serving, which is a lightweight distributed, real-time serving solution that supports a wide range of deep learning models (such as TensorFlow, PyTorch, Caffe, BigDL and OpenVINOTM models). As shown below, Analytics Zoo/BigDL Cluster Serving components include a web front end, Redis, an inference engine (e.g., TensorFlow or OpenVINOTM), and distributed streaming frameworks (e.g., Apache Flink). The inference engine, streaming frameworks, web front end and Redis run on top of Graphene and inside Intel SGX enclaves, meanwhile web front end and Redis are also TLS-protected or encrypted, protecting the user data and model in the inference pipeline when in-store, in-transit or in-use.

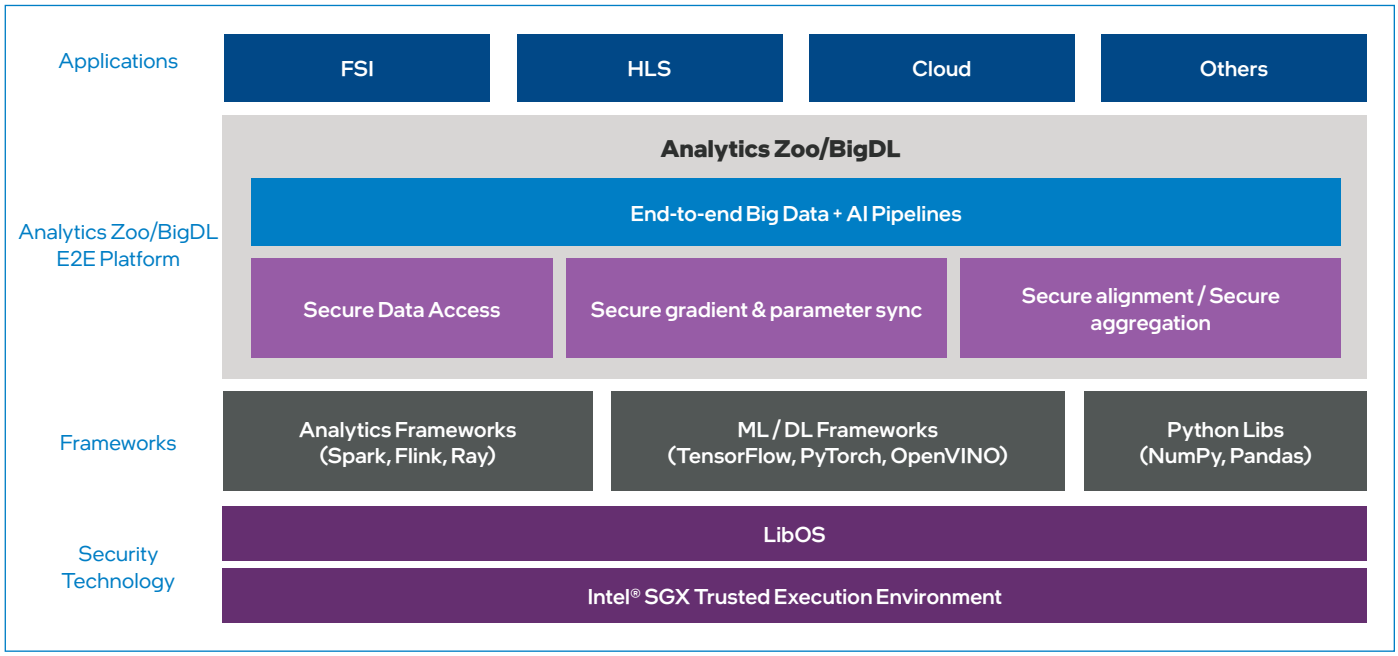

Figure 4. Analytics Zoo/BigDL PPML

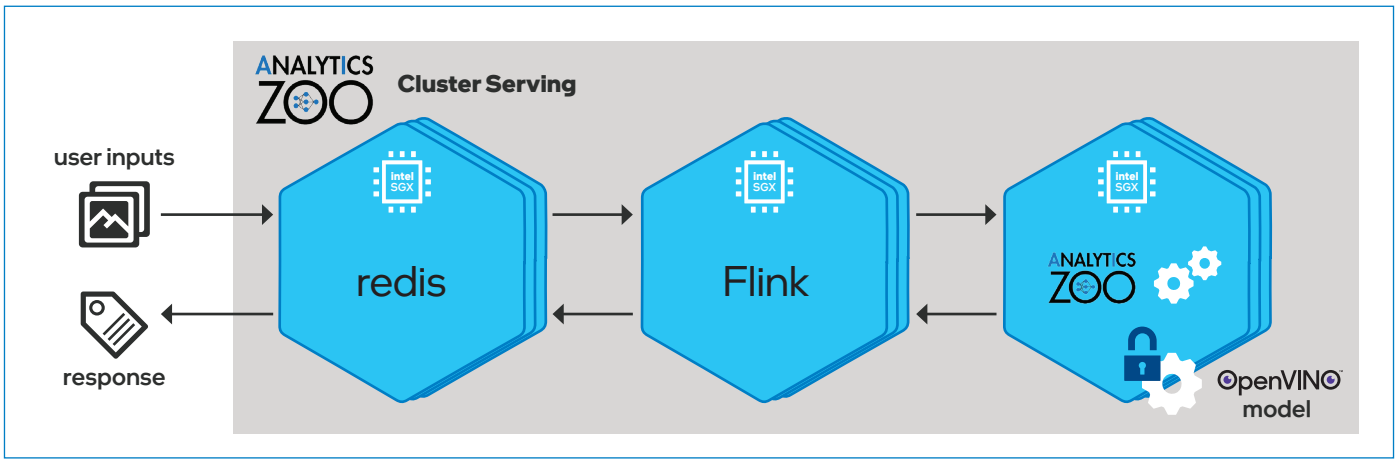

Figure 5. Analytics Zoo/BigDL PPML – Trusted Cluster Serving with SGX

1.3.4 Intel 3rd Gen Xeon Scalable Platform SKUs for Enclave Page Cache

The 3rd Gen Intel® Xeon® Scalable Platform (SP) SKUs listed below set the maximum Enclave Page Cache (EPC) sizes as shown below. The BIOS configures the desired EPC size in powers of 2 (e.g., 1 GB, 2 GB, 4 GB, 8 GB, 16 GB, etc.) up to the maximum capacity setting. Non-EPC memory space must be equal or larger than allocated EPC space.

- Platinum 83XX (2S) SKU Max EPC (Enclave Page Cache) = 512 GB + 512 GB= 1 TB Total
- Gold 63XX (2S) SKU Max EPC (Enclave Page Cache) = 64 GB + 64 GB = 128 GB Total
- Silver 43XX (2S) SKU Max EPC (Enclave Page Cache) = 8 GB + 8 GB = 16 GB Total

2. Test Setup

2.1 Device Under Test Configuration

Table 2. shows the overall hardware and software composition of the device under test (DUT) and the application container configuration. It was configured to use SGX for each Redis test pod (Redis, memtier), PPML test pod (Flink JobManager, Flink TaskManager, JobMaster) by installing the SGX device plugin as a Kubernetes daemon set. SGX can then be discovered from nodes through node-feature-discovery (NFD). The SGX device plugin daemon set will be deployed to the nodes where SGX is discovered as separate namespaces. The SGX enclave was configured to use 256 GB total (128 GB + 128 GB) when the BIOS Processor Reserved Memory (PRM) was set up. The kernel was built with the configuration shown in Table 2.

Table 2: DUT Configuration

Category		Description
Processor	Product	Intel® Xeon® Gold 8352S Processor
	Frequency	2.2 to 3.4 GHz
	Cores per processor	32 cores, 64 hyper-threads
Memory	DIMM slots per processor	8 channels per processor
	Capacity	512 GB DRAM (32 GB x16 DIMM)
	Memory speed	3200 MHz (MT/s), DDR4
	SGX EPC	256 GB, 128 GB per socket
Network	Number of ports	2-port X550-T
Storage	Vendor	1x Intel S4610 960 GB SSD INTEL_SSDSC2KG96
2RU server	Vendor	Intel M50CYP WHITLEY
Host OS	Vendor/version	Ubuntu 20.04 LTS with Linux 5.13.4 x86_6
BIOS	Vendor/version	Intel Corporation Version: SE5C6200.86B.0022.D64.2105220049 Release Date: 05/22/2021
Microcode	Vendor/version	Intel Corporation microcode: 0xd0002b1
Kubernetes	Vendor/version	Opensource, v1.22.1 (client & server)
SGX device plugin	Vendor/version	Intel, v0.21.0 https://github.com/intel/intel-device-plugins-for-kubernetes.git
Redis test pod	Vendor/version	Opensource, Redis v=6.0.5 on Ubuntu 18.04 LTS
Library OS	Vendor/version	Opensource, Graphene v1.2-rc1 https://github.com/oscarlab/graphene.git
PPML test pod	Vendor/version	Oracle Java 1.8.0_192 Opensource Python 3.6.9 Redis 6.0.5 Apache Flink 1.11.3 Analytics Zoo/BigDL 0.12 snapshot OpenVINO 2020.3.2

Figures 6 and 7 show the overall structure of the system architecture and Redis encryption with SGX to validate various SGX-enabled confidential computing services on top of SKT MEC. In order to isolate any inconsistency that occurred in each of the test results, separate CPU cores were assigned to each Redis and PPML pod.

A Redis container was used to measure SGX overhead by GET/SET OPS (operations per second) results through the memtier benchmark tool. Two vCPUs each with 2 GB memory were assigned for Redis Origin and Redis Direct. Two vCPUs each with 2 GB memory and SGX EPC were allocated for the SGX and SGX-exitless containers. Four memtier containers were deployed to make a corresponding Redis pair as shown below. Two vCPUs each with 1 GB memory were assigned to each memtier pod as well (see Listing 1).

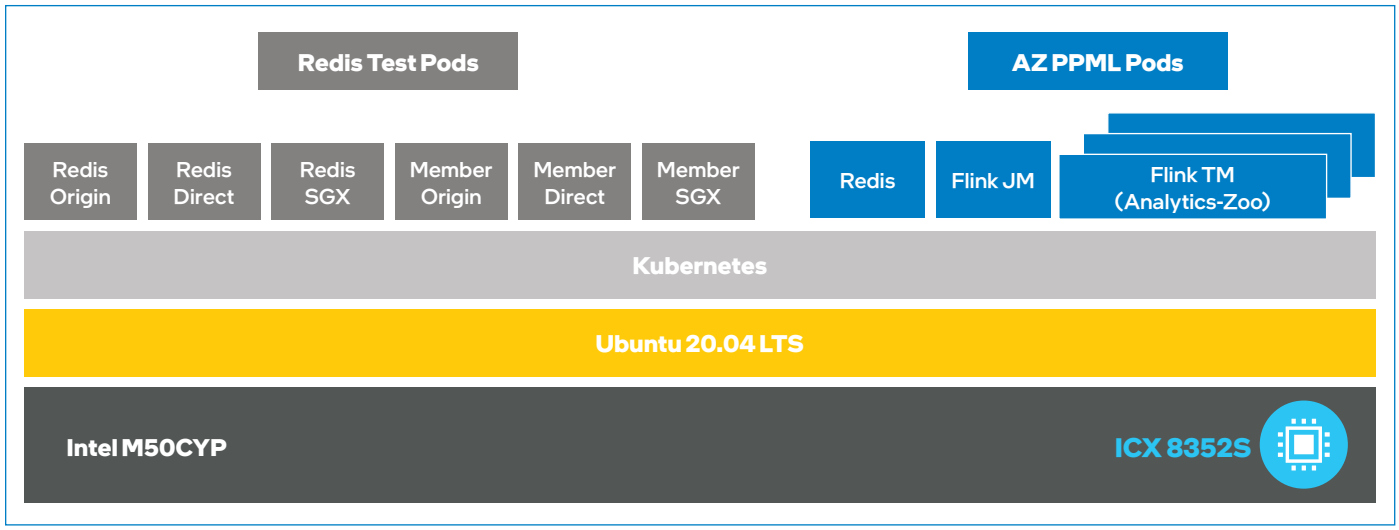

Figure 6. Systems Architectures

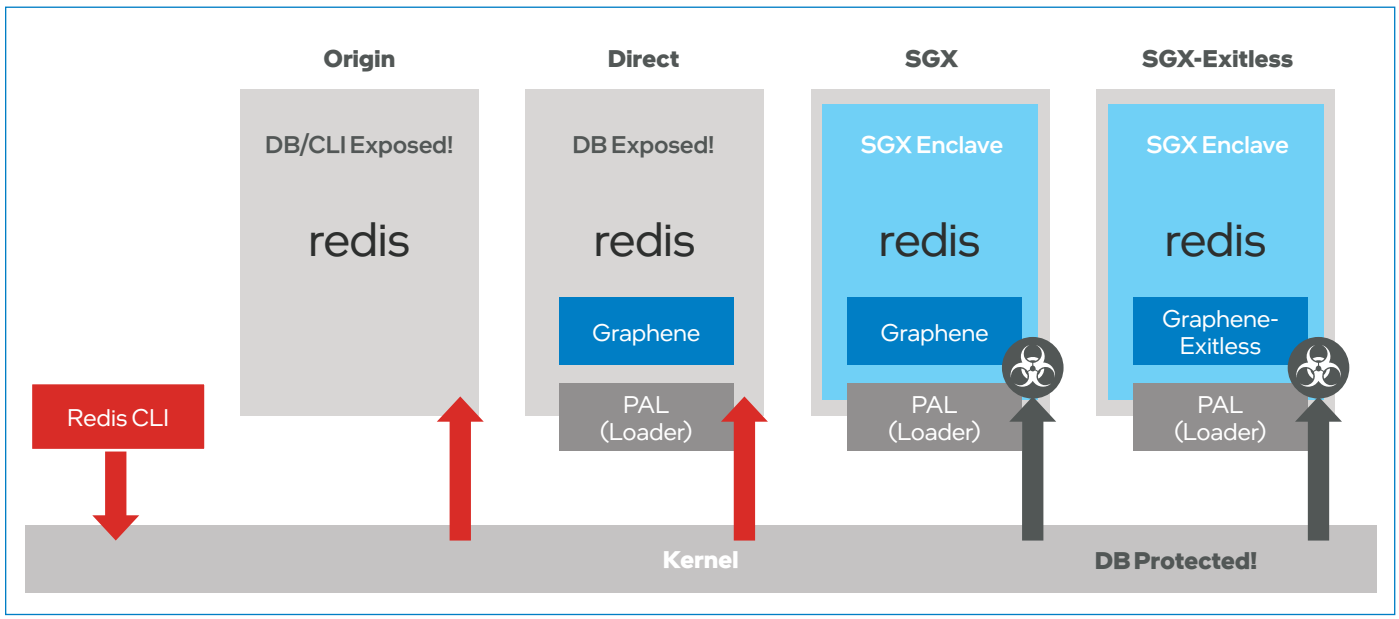

Figure 7. Redis Encryption with SGX

Listing 1: Redis PoD Configuration

```
intel@icx-sdp4:~/multi_Redis$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
memtier-5b99b5958d-25gxg            1/1     Running   0          2d3h
memtier-5b99b5958d-5k27x            1/1     Running   0          2d3h
memtier-5b99b5958d-5rm5s            1/1     Running   0          2d3h
memtier-5b99b5958d-ql6f2            1/1     Running   0          2d3h
redis-direct-66756f8575-xltfj       1/1     Running   0          2d3h
redis-exitless-5b8cdc6595-drrwg     1/1     Running   0          2d3h
redis-origin-54d8bcfb48-vvn29       1/1     Running   0          2d3h
redis-sgx-5d754547bb-wdkqn          1/1     Running   0          2d3h

intel@icx-sdp4:~/multi_Redis$ kubectl get pods redis-origin-54d8bcfb48-vvn29  --output=yaml
apiVersion: v1
-
spec:
 containers:
 - image: Redis_sgx:0.1
   imagePullPolicy: Never
   name: direct
   resources:
     limits:
       cpu: "2"
       memory: 2Gi
     requests:
       cpu: "2"
       memory: 2Gi

intel@icx-sdp4:~/multi_Redis$ kubectl get pod redis-exitless-5b8cdc6595-drrwg --output=yaml
-
spec:
 containers:
 - image: Redis_sgx:0.1
   imagePullPolicy: Never
   name: exitless
   resources:
     limits:
       cpu: "2"
       memory: 1Gi
       sgx.intel.com/enclave: "1"
       sgx.intel.com/epc: 2Gi
     requests:
       cpu: "2"
       memory: 1Gi
       sgx.intel.com/enclave: "1"
       sgx.intel.com/epc: 2Gi
```

The major components for PPML are one Flink JobManager pod, multiple Flink TaskManager pods, and one JobMaster pod. The Redis server/client also resides in a JobMaster pod. A Flink JobManager pod is allocated with two vCPUs each with 16 GB memory and 16 GB SGX EPC. Each Flink TaskManager pod is allocated with 8 vCPUs, each with 16 GB memory and 16 GB SGX EPC. A JobMaster pod is allocated with 12 vCPUs each with 16 GB memory and 16 GB SGX EPC. The pod configurations are listed in Listing 2 and Figure 6.

Listing 2: PPML PoD Configuration

intel@icx-sdp4:~$ kubectl get pods

NAME	READY	STATUS	RESTARTS	AGE
flink-jobmanager-57884b6577-c6npv	1/1	Running	0	18h
flink-taskmanager-0	1/1	Running	0	21h
flink-taskmanager-1	1/1	Running	0	21h
flink-taskmanager-2	1/1	Running	0	21h
flink-taskmanager-3	1/1	Running	0	21h
flink-taskmanager-4	1/1	Running	0	21h
flink-taskmanager-5	1/1	Running	0	21h
flink-taskmanager-6	1/1	Running	0	21h
flink-taskmanager-7	1/1	Running	0	21h
master-deployment-78f7479cf5-jwjpt	1/1	Running	0	8d

2.2 Test Procedure

2.2.1 Redis-SGX Performance

To demonstrate the benefit of Intel SGX, we tested Redis container performance with DB key encryption. The performance tests used the memtier benchmark tool with automation provided by Intel-developed scripts that automatically start the memtier benchmark tools. The following Redis pod was measured to compare performance.

- Average operations per millisecond of an instantiated Redis-SGX container:

  memtier-5b99b5958d-25gxg to Redis-origin-54d8bcfb48-vvn29 (10.245.3.15)

  memtier-5b99b5958d-5k27x to Redis-direct-66756f8575-xltfj (10.245.3.16)

  memtier-5b99b5958d-5rm5s to sgx-5d754547bb-wdkqn (10.245.3.17)

  memtier-5b99b5958d-ql6f2 to Redis-exitless-5b8cdc6595-drrwg (10.245.3.18)

- The memtier benchmark tool configuration was set to a data size of 2048 bytes, 4:1 GET/SET ratio with Gaussian key-pattern, and 32 connections per thread. The results were measured as a SKT SLA condition as shown below, 10 times iteration (warm) and the average resulting OPS was chosen as a result.

  memtier_benchmark options: --ratio=1:4 -d 2048 --key-pattern=G:G --key-minimum=1 --key-maximum=50001 --threads=1 --pipeline=1 -c 32 --hide-histogram --test-time=100

- To test SGX encryption functionality, the Redis set key was sent by Redis-cli. Key/value means that redis DB key/value paires to test encyption and 10ea key/value was used from test00/intel00 to test09/intel09. The process was then dumped by gcore for process PID then saved as a core_dump.PID (binary file), (using gcore -o core_dump PID). If the DB is in an SGX enclave, its key value will not be printed. If it is not in enclave, it will be printed on the screen.

2.2.2 PPML SGX Performance

The performance of PPML cluster serving is measured by end-to-end throughput (images per second from image enqueue to inference result dequeue), taking the average of 15 rounds with 10000 images per round. The benchmark script used a client Python API of cluster serving to enqueue test images and dequeue inference results. The test workload is a Resnet50 model inference with the OpenVINO engine. Batch size is 32 per instance, and 256 in total. To further improve E2E PPML cluster serving performance with Graphene-SGX, exitless mode is enabled for Redis.

2.2.3 SGX Integration with SKT 5G MEC Platform

To see interoperability between SGX and SKT Edge Stack, an SGX-Redis test container was instantiated through the SKT MEC Orchestrator. The instantiation status was then monitored whether it worked and optimized.

3. Test Result

3.1 Redis-SGX Performance

For the Redis performance comparison, the tests measured overall average throughput in operations per second of every container of Origin, Direct, SGX, and SGX exitless that utilized two vCPUs on the same physical core. Each performance was compared to that of the Origin to measure overhead. Figure 8 provides Redis-SGX performance for each Redis container under 2048 bytes, Test highlights include:

- SGX overhead was 64 percent in SGX and 25 percent in exitless. Graphene LibOS also used 17 percent overhead without SGX.
- The SGX exitless (switchless) feature provided 208 percent improvement in OPS and 52 percent improvement in latency overhead by additional vCPU. EPC is used for SGX and exitless Redis, so pod memory usage is not increased for them (see Figure 9).
- Demonstrated that the Redis DB key was protected under SGX enclave as shown below. The last key and value can be printed because the platform adaptation layer (PAL) is not running in an enclave and has a packet buffer for the last CLI command; it is not from the DB. If a TLS connection is used, E2E encryption will be supported (see the listings in Table 3).

Table 3: Redis key encryption

intel@icx-sdp4:~/multi_Redis$ ./dump_test.sh origin [Inferior 1 (process 67782) detached] test09 intel09 - test01 intel01 test00 intel00	intel@icx-sdp4:~/multi_Redis$ ./dump_test.sh sgx [Inferior 1 (process 68870) detached] *test09 *intel09 - xxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxx
intel@icx-sdp4~/multi_redis$ ./dump_test.sh sgx tls Test with TLS connnection - [New LWP 395545] 0x000055fc36c0356c in sgx_ocall_poll () Saved corefile core_dump.395494 [Inferior 1 (process 395494) detached]	intel@icx-sdp4:~/multi_Redis$ ./dump_test.sh exitless [Inferior 1 (process 1300729) detached] *test09 *intel09 - xxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxx

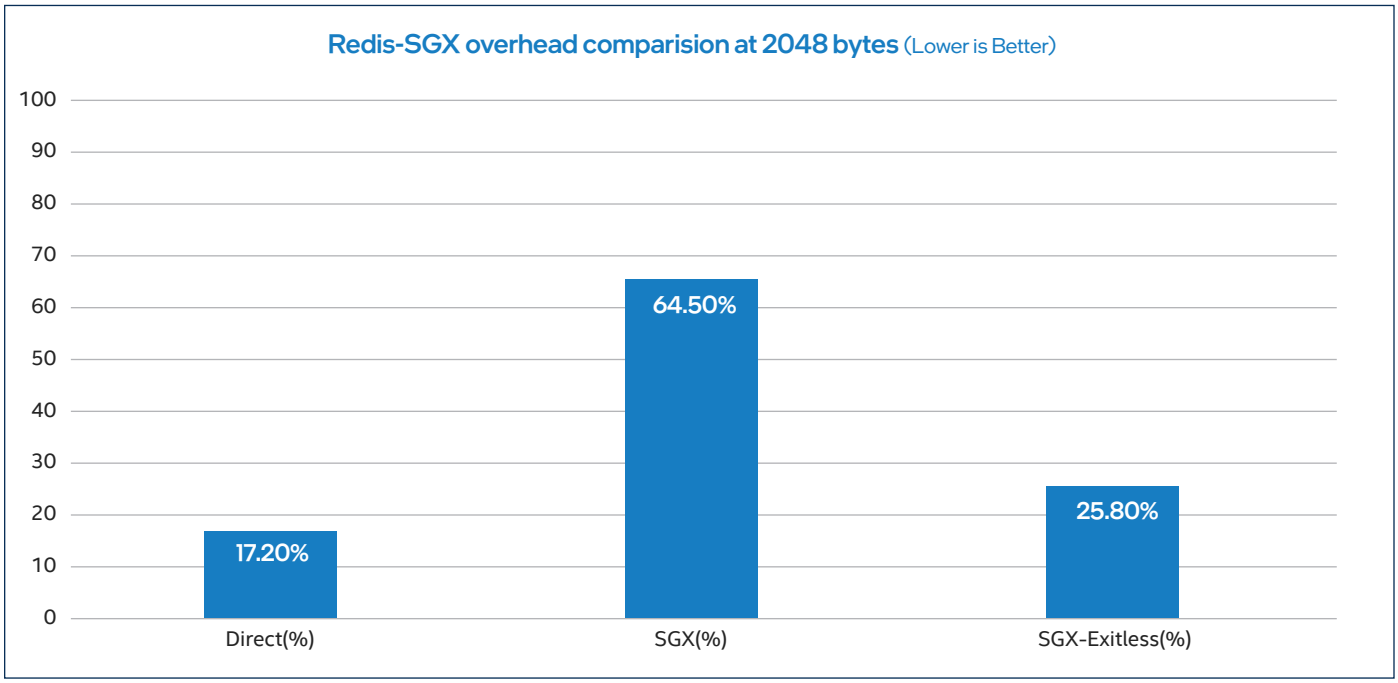

Figure 8. Redis-SGX performance at 2048 bytes

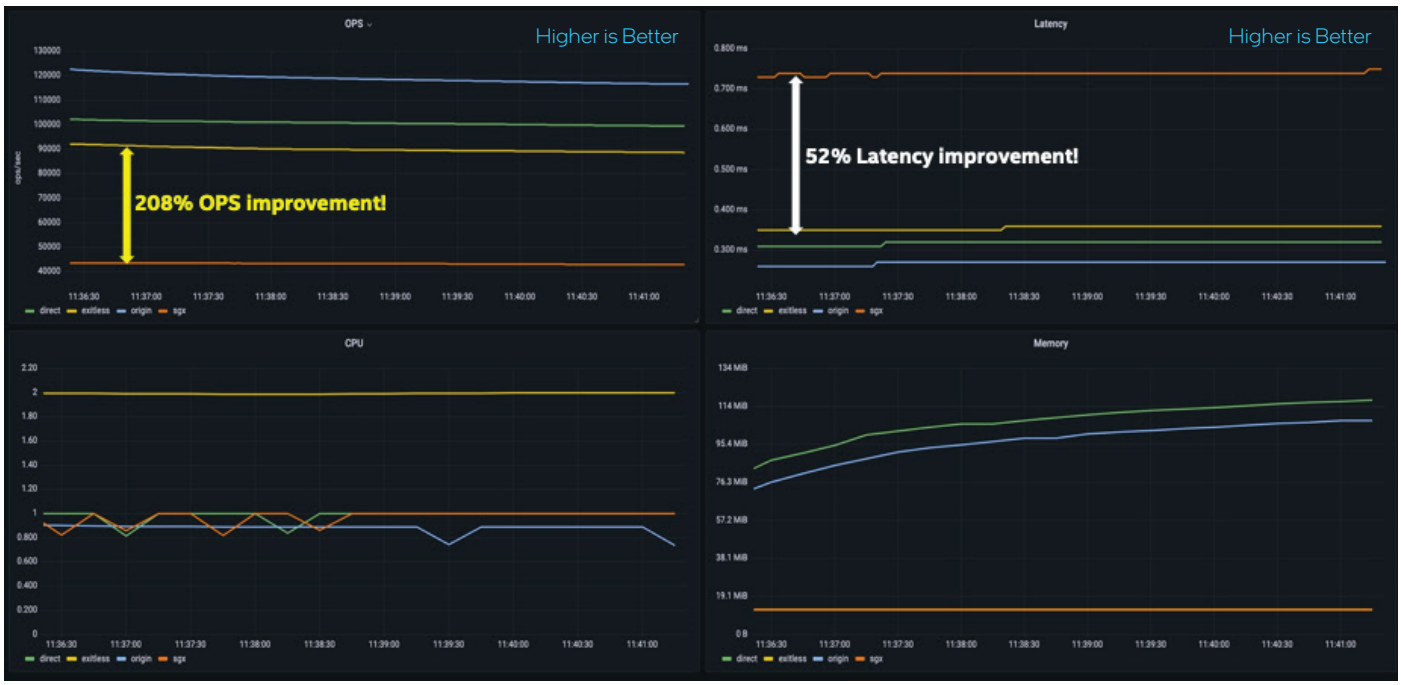

Figure 9. Redis-SGX performance at 2048 bytes

3.2 PPML-SGX Performance

The performance test for PPML compared the end-to-end cluster serving throughput of non-SGX mode and graphene-SGX mode for precision FP32 and INT8. In the test results shown in Figure 10, the Graphene-SGX overhead is 4.8 percent for FP32 and 14.4 percent for INT8.

3.3 SGX Integration with SKT MEC Platform

Figures 11 and 12 show that when confidential computing service is officially commercialized at SKT MEC, customers will be able to use the service directly through the SKT MEC Biz. Console. It also shows that the resource status of Redis test pods can be continuously monitored through the SKT MEC Orchestrator's assurance window, such as real time CPU usage and memory usage.

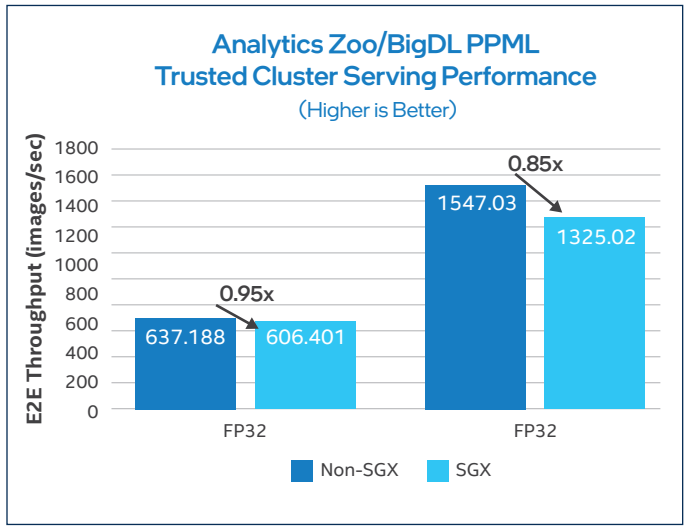

Figure 10. Analytics Zoo/BigDL PPML Trusted Cluster Serving Relative Performance

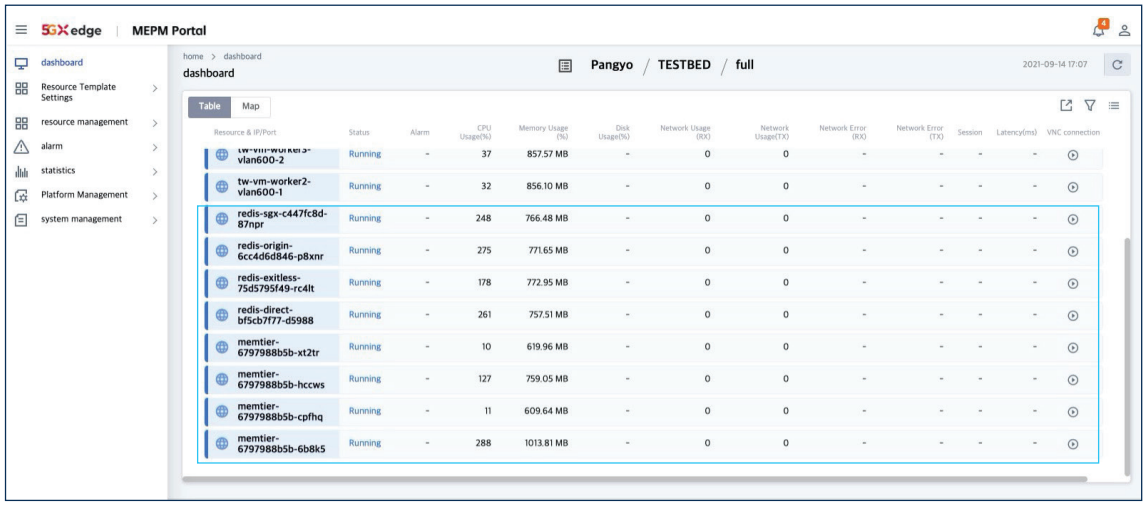

Figure 11. SKT MEC Management Portal

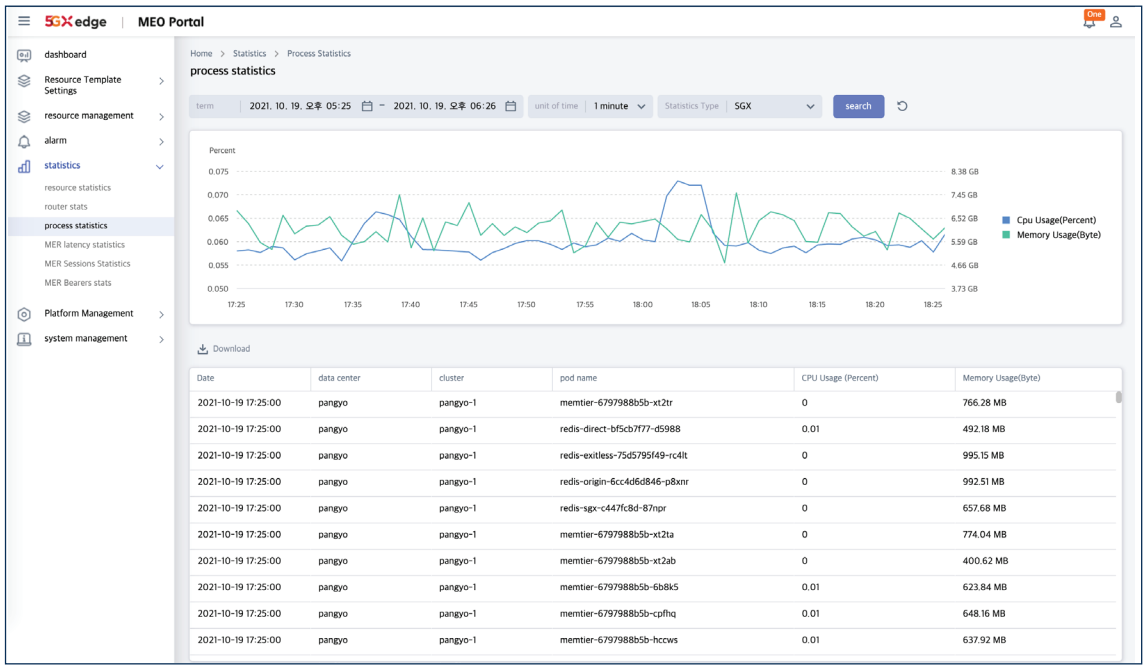

Figure 12. SKT MEC statistics

4. Summary

These tests present a confidential computing reference architecture to validate interoperability between Intel SGX and SKT MEC edge stack for commercial services, leveraging Intel 3rd Xeon SP's enhanced hardware security features. The tests described in this paper include:

- Proven interoperability between Intel SGX and SKT 5G 5G MEC edge stack
- Demonstrated confidential computing as a service through SGX-enabled Redis, and PPML container performance validation on SKT 5G MEC edge stack

The above results indicate that telecom equipment manufacturers (TEMs) and independent software vendors (ISVs) can utilize SGX to help meet the requirements for confidentiality and integrity on open platforms being driven by rising robustness rules for content protection on edge computing services.

The mobile network operator can leverage this confidential computing reference architecture to meet customer demand for new services such as block chain, secret storage, ML-inferencing IoT, data integrity, privacy-preserving machine learning, federated learning, etc. with data integrity and confidentiality to help manage B2B and B2B2C sales revenues.

5. Resources

Confidential Computing
https://www.intel.co.kr/content/www/kr/ko/now/edge-to-cloud/confidential-computing-case-study.html

Intel 3rd Gen Xeon Scalable Processor
https://www.intel.com/content/www/us/en/products/docs/processors/xeon/3rd-gen-xeon-scalable-processors-brief.html

Intel SGX
https://software.intel.com/content/www/us/en/develop/topics/software-guard-extensions.html

Graphene
https://graphene.readthedocs.io/en/latest/index.html

Graphene Exitless Feature
https://graphene.readthedocs.io/en/latest/devel/performance.html?highlight=exitless#effects-of-system-calls-ocalls

Intel SGX Device Plugin for Kubernetes
https://intel.github.io/intel-device-plugins-for-kubernetes/cmd/sgx_plugin/README.html

Kubernetes Node Feature Discovery
https://docs.01.org/kubernetes/nfd/overview.html

Kubernetes Topology Manager
https://kubernetes.io/docs/tasks/administer-cluster/topology-manager/

Redis Benchmark Tool
https://github.com/RedisLabs/memtier_benchmark

Intel PPML and Analytics Zoo/BigDL
https://www.intel.com/content/www/us/en/artificial-intelligence/posts/alibaba-privacy-preserving-machine-learning.html

https://analytics-zoo.readthedocs.io/en/latest/doc/PPML/Overview/ppml.html

6. Glossary of Terms

Term	Description
SGX	Software Guard Extension
K8S	Kubernetes
NFV	Network Function Virtualization
NFVi	Network Function Virtualization Infrastructure
EPC	Enclave Page Cache
PRM	Processor Reserved Memory
BIOS	Basic Input/Output System
NFD	Node Feature Discovery
MEC	Mobile Edge Computing
TEM	Telecom Equipment Manufacturer
ISV	Independent Software Vendor
CNF	Cloud Native network Function
TLS	Transport Layer Security
PPML	Privacy Preserving Machine Learning
PAL	Platform Adaptation Layer

7. Authors

SK Telecom Head of 5GX MEC Product, 5GX Intelligence Company, T3K (ICT R&D Center)

Moonyoung Chung,
Solution Architect,mychung@sk.com

Daniel Park,
Solution Architect, dan.park@sk.com

Jian Li,
Solution Architect, gunine@sk.com

Keunhyun Kim,
Product Manager, keunhyun.kim@sk.com

Intel Corporation Data Platform Group

Wooram Alex Kim,
NCS Solutions Architect, alex.kim@intel.com

Jongeop Jayden Lee,
NPG PAE, jayde.lee@intel.com

Intel Corporation IAGS

Qiyuan Gong,
qiyuan.gong@intel.com

Dongjie Shi,
dongjie.shi@intel.com

Yabai Hu,
yabai.hu@intel.com

Tests conducted by Intel in Sep. 2021:

The server was an Intel® Server System Intel M50CYP WHITLEY by a 2.6 GHz Intel Xeon Gold 8352S Processor (microcode 0xd0002b1). The server featured 1 nodes and 2 sockets. Both Intel® Hyper-Threading Technology and Intel® Turbo Boost Technology were turned on. Total memory equaled 512 GB DRAM (32 GB x16 DIMM), BIOS version is SE5C6200.86B.0022.D64.2105220049

Ubuntu 20.04 LTS with Linux 5.13.4 x86_6 was the operating system, SKT Edge stack was used for MEC SW and Redis 6.0.5 on Ubuntu 18.04 LTS container. k8s: v1.22.1 (Client & Server), SGX device plugin: v0.21.0, Graphene: v1.2-rc was used. The Server ran memtier benchmark tool configuration was set to a data size of 2048 bytes, 4:1 GET/SET ratio with Gaussian keypattern, and 32 connections per thread. The results were measured as a SKT SLA condition as shown below, 10 times iteration (warm) and the average resulting OPS was chosen as a result and the performance of PPML cluster serving is measured by end-to-end throughput (images per second from image enqueue to inference result dequeue), taking the average of 15 rounds with 10000 images per round. The benchmark script used a client Python API of cluster serving to enqueue test images and dequeue inference results. The test workload is a Resnet50 model inference with the OpenVINO engine. Batch size is 32 per instance, and 256 in total. To further improve E2E PPML cluster serving performance with Graphene-SGX, exitless mode is enabled for Redis.

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.

For workloads and configurations visit www.Intel.com/PerformanceIndex

Results may vary

Intel technologies may require enabled hardware, software, or service activation.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

1021/BB/MESH/PDF 349047-001US