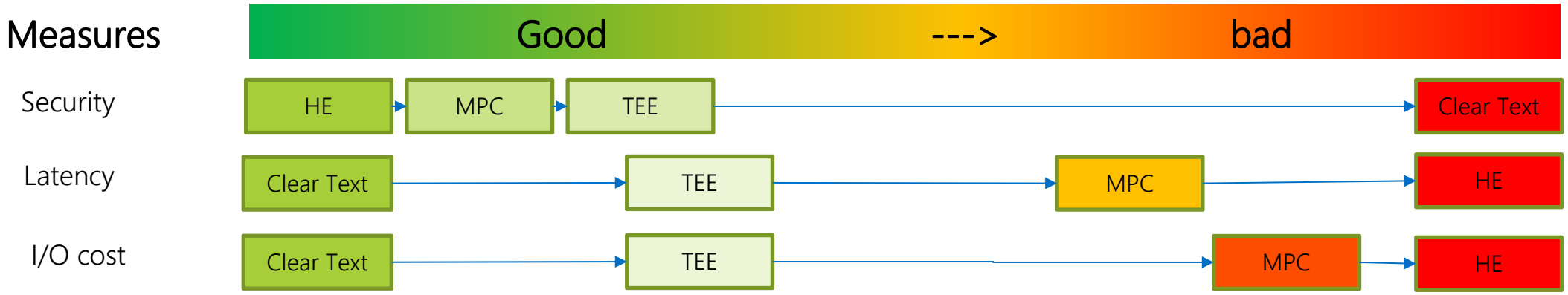
Analytics Zoo and PPML

Dongjie Shi

Agenda

- TEE
- Intel SGX
- Graphene-SGX
- Analytics Zoo
- Analytics Zoo Secured Cluster Serving on Graphene-SGX

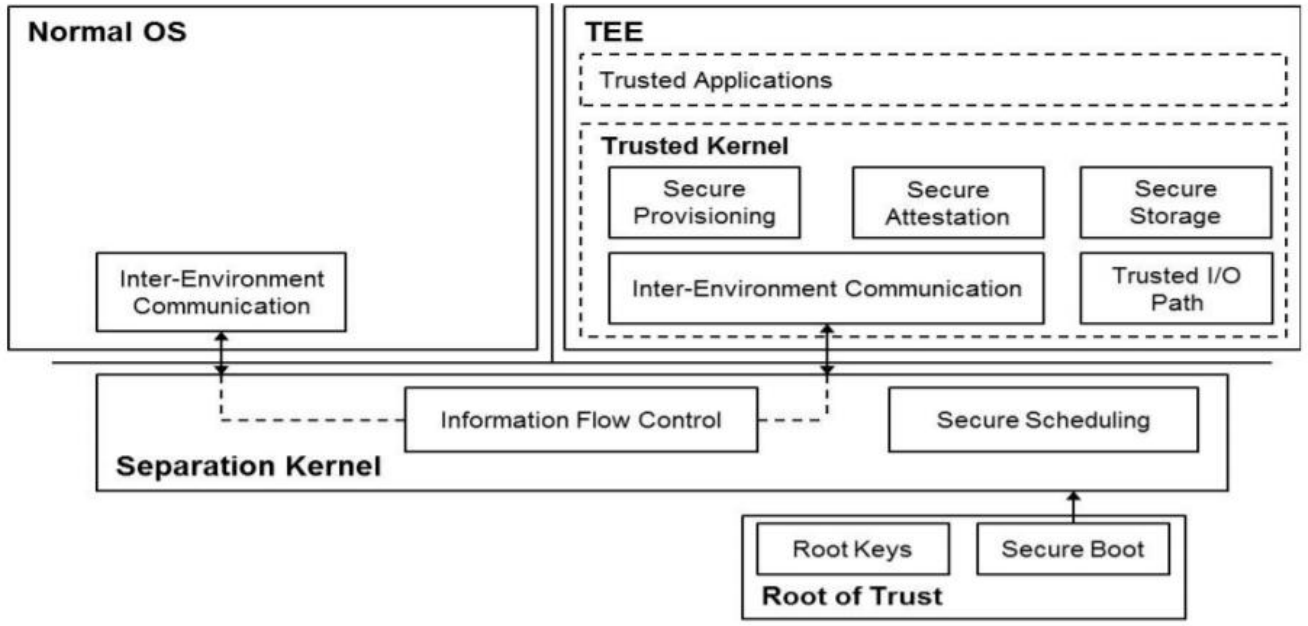
Comparison of PPML Technologies



PPML Technologies	TEE	MPC	HE	Clear Text
Implementation	CPU HW instruction sets + SW SDK	dominated by GPU acceleration	Current CPU dominate, but with GPU optimization	CPU/GPU
Computation Speed	10%~10X slower	10~100X slower	~10000x slower	1x (Baseline)

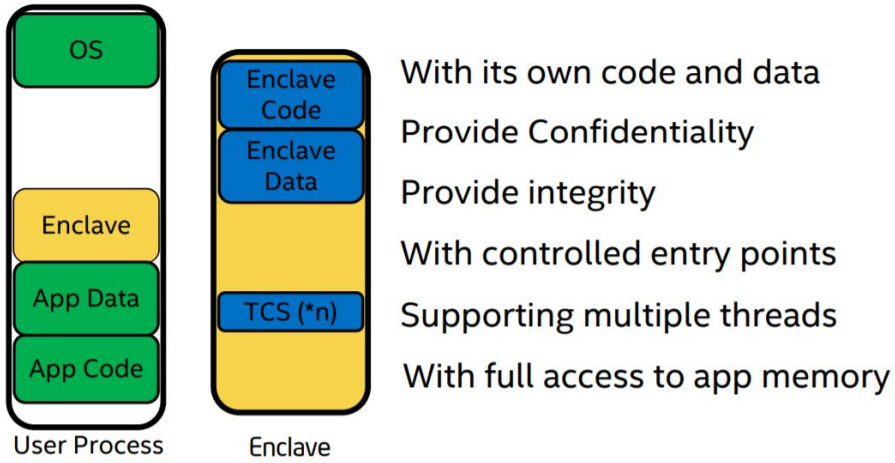
TEE: Trusted Execution Environment

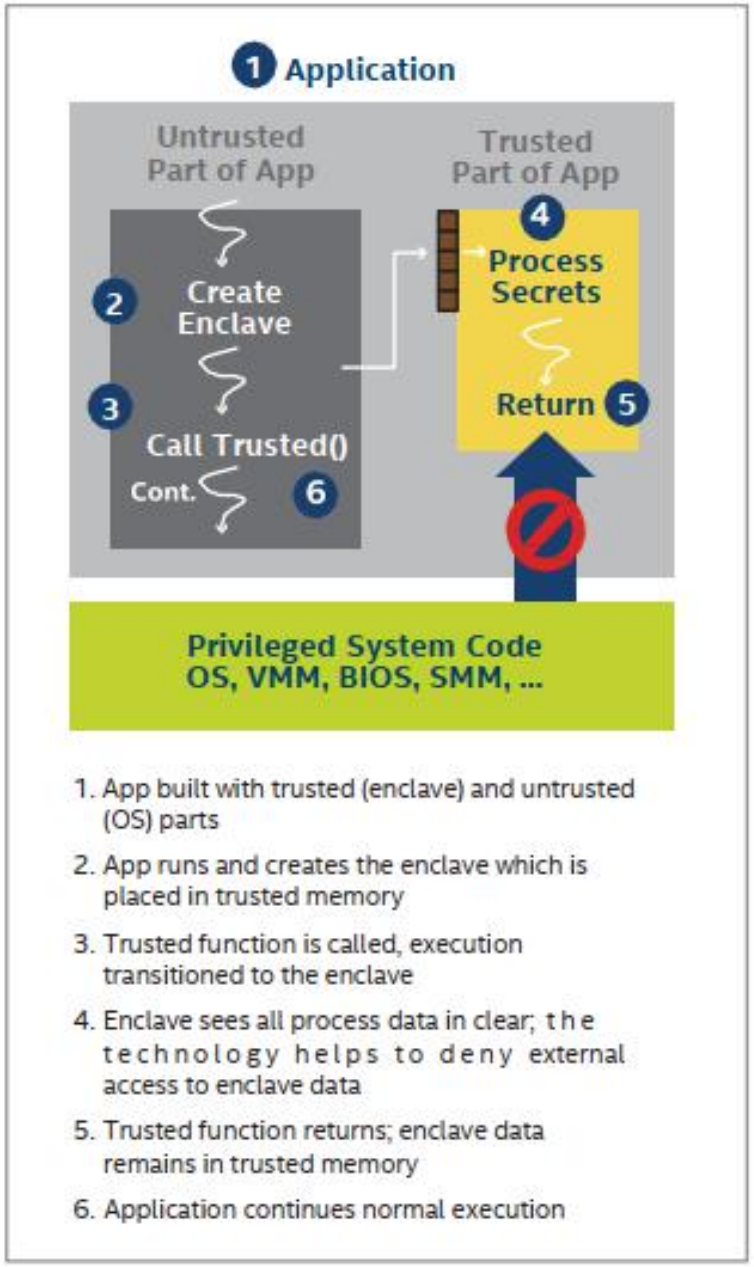
- TEE is a tamper resistant processing environment that runs on a separation kernel.
- Goals of TEE
  - **Isolated Execution**
    - TEE/Normal OS may be malicious
  - **Secure Storage**
    - Integrity, Confidentiality, Freshness
  - **Remote Attestation**
    - determine the level of trust in the integrity of attestator
  - **Secure Provisioning**
    - remotely manage and update its data in a secure way
  - **Trusted I/O Path**
    - protects authenticity, and optionally confidentiality, of communication between TEE and peripherals



Intel® SGX (Intel® Software Guard Extensions)

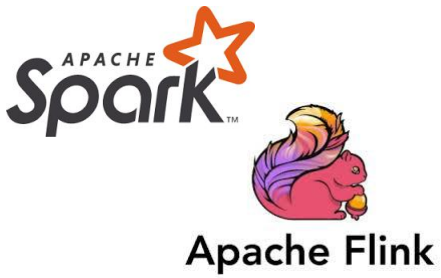
- SGX protects selected code and data from disclosure or modification.
  - Enhances confidentiality and integrity
  - Low learning curve
  - Remotely attest and provision
  - Help Significantly reduce attack surface
- The primary SGX abstraction is an enclave: an isolated execution environment within the virtual address space of a process.



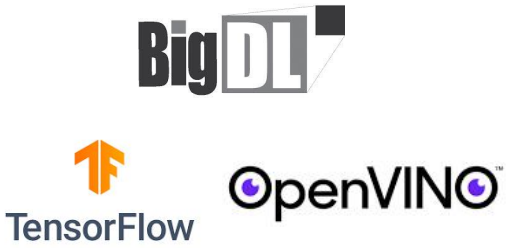


Graphene-SGX:
A Practical Library OS for **Unmodified** Applications on SGX

Case Studies



Big Data



AI

Big Data + AI

Properties
- *Mature, widely deployed code*
- *Complex with millions of lines of code*
- *Written in C, Python, Java*
- *Use OS services*

Refactoring Challenges
- *Takes time and effort*
- *May introduce bugs in mature code*
- *Need a separate SDK for every language*
- *Developer expertise to know how to refactor*
- *No access to source code of 3rd party apps*

intel Software

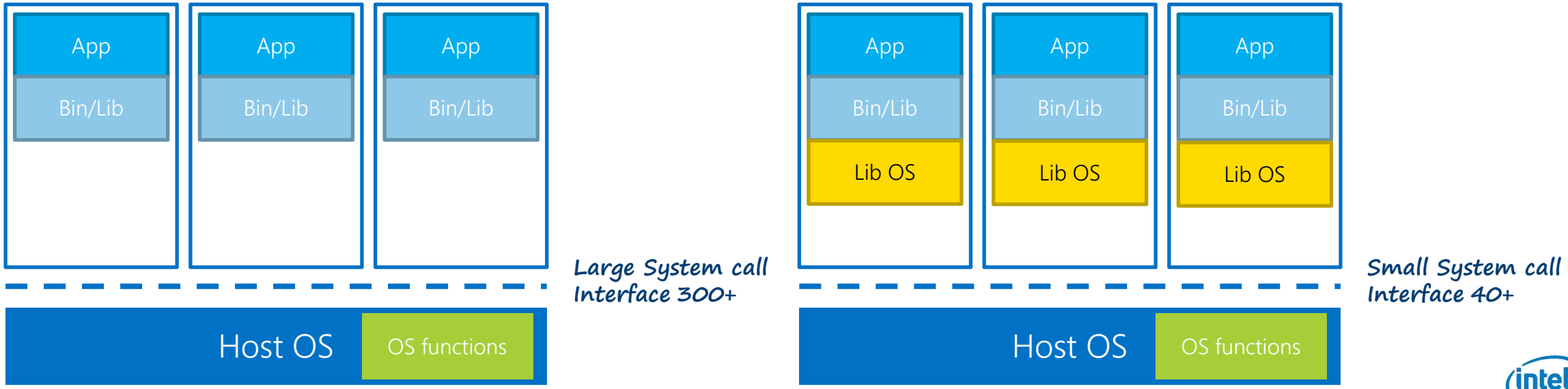
*Other names and brands may be claimed as the property of others.

Graphene-SGX: Objectives & Solution

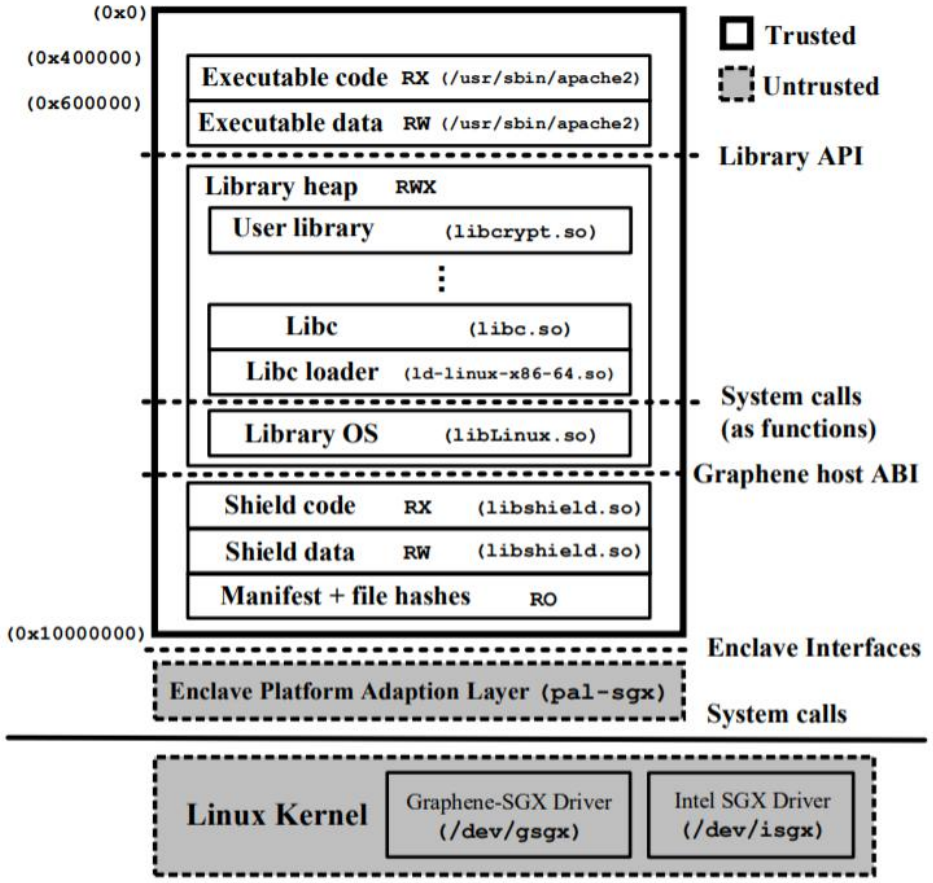
- Objectives
  - **Develop a tool to secure an application that:**
    - Provides isolation without any code modification
    - Supports policy-based security enforcement (Manifest, trusted, allowed)
    - Does not compromise performance
    - Provides transparent support for attestation and secure OS services
- Solution
  - **Graphene:** A Library OS for running unmodified Linux applications inside SGX enclaves

Library OS Background

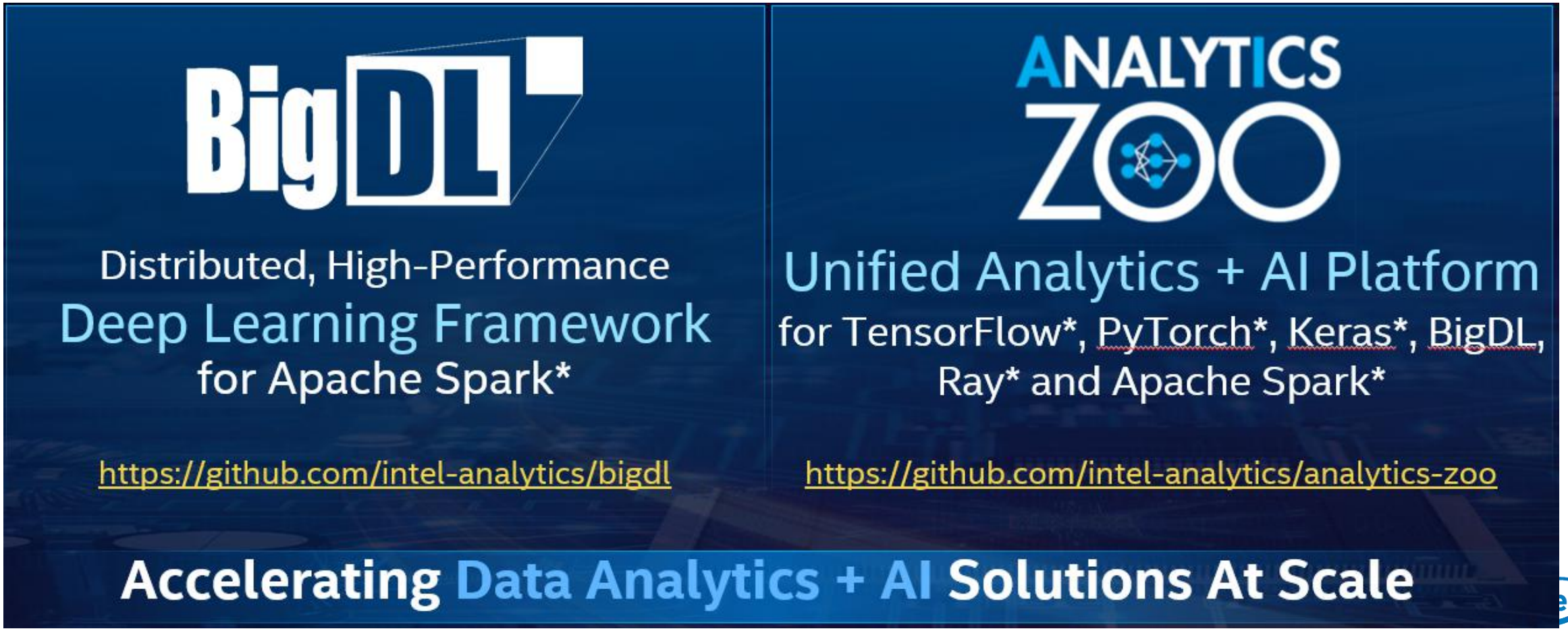
- Approach was championed by several OS designs in early 90's with a focus on performance
- Runs a part of OS functionality as a library in application
- Communication to the host OS with a small fixed set of abstractions
- Offers a secure alternative to large traditional OS system call interface



Graphene-SGX Architecture



AI on Big Data

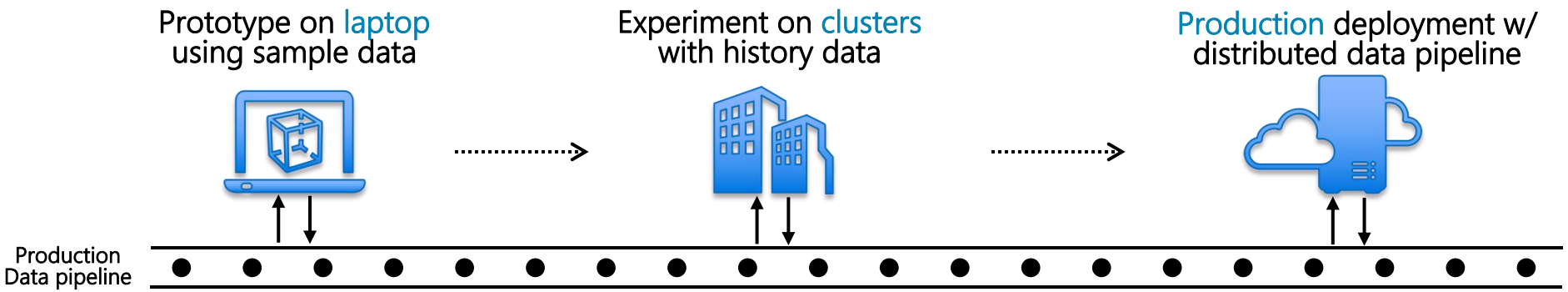


Software

*Other names and brands may be claimed as the property of others.

Integrated Big Data Analytics and AI

Seamless Scaling from Laptop to Distributed Big Data

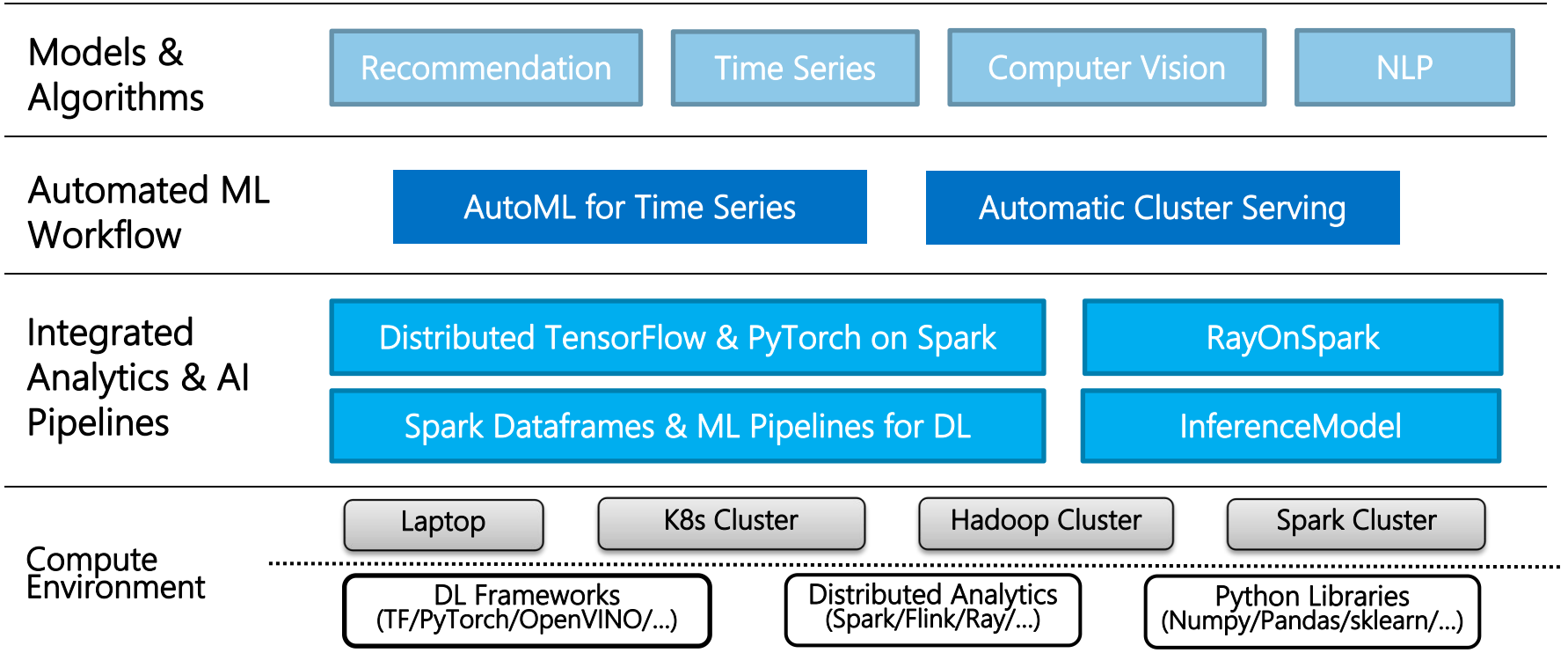


- Easily prototype end-to-end pipelines that apply AI models to big data
- "Zero" code change from laptop to distributed cluster
- Seamlessly deployed on production Hadoop/K8s clusters
- Automate the process of applying machine learning to big data

intel Software

Analytics Zoo

Unified Data Analytics and AI Platform

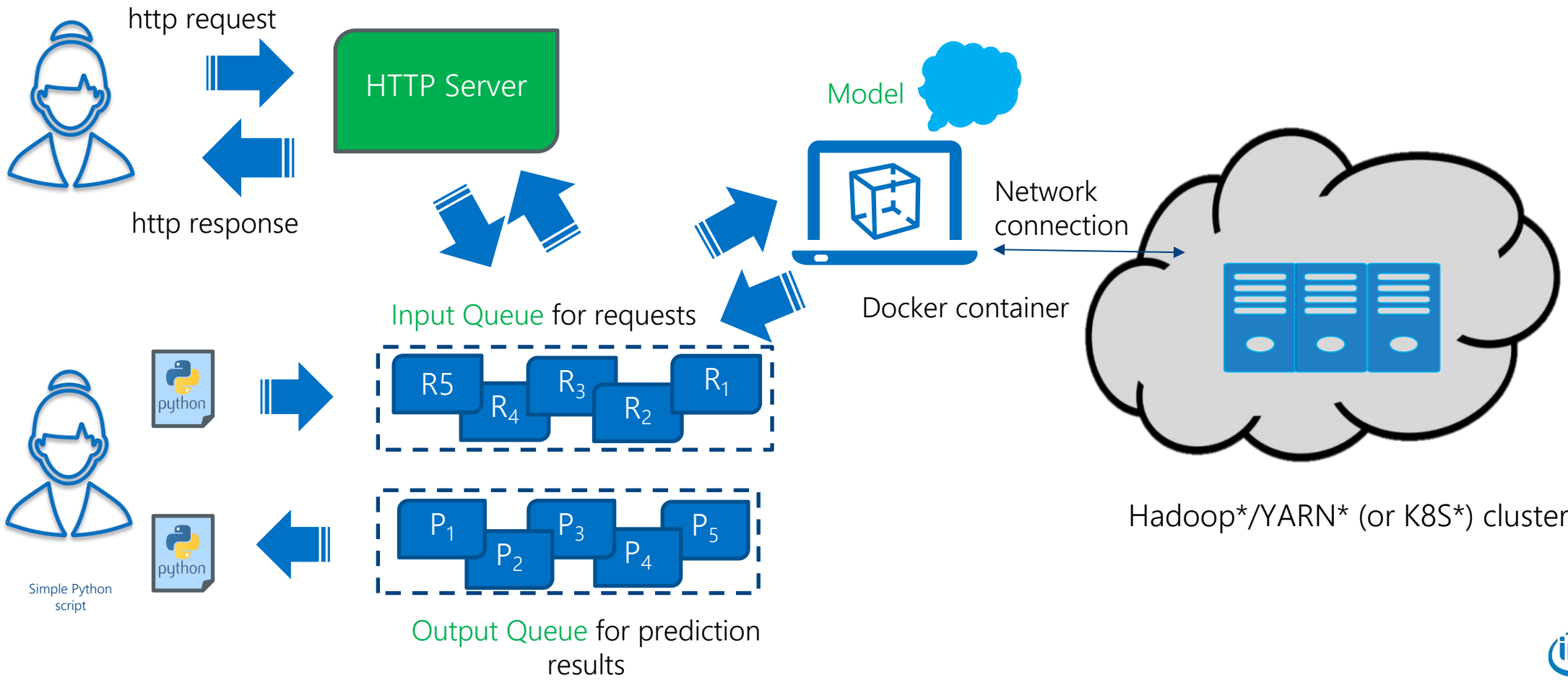


Powered by oneAPI

https://github.com/intel-analytics/analytics-zoo

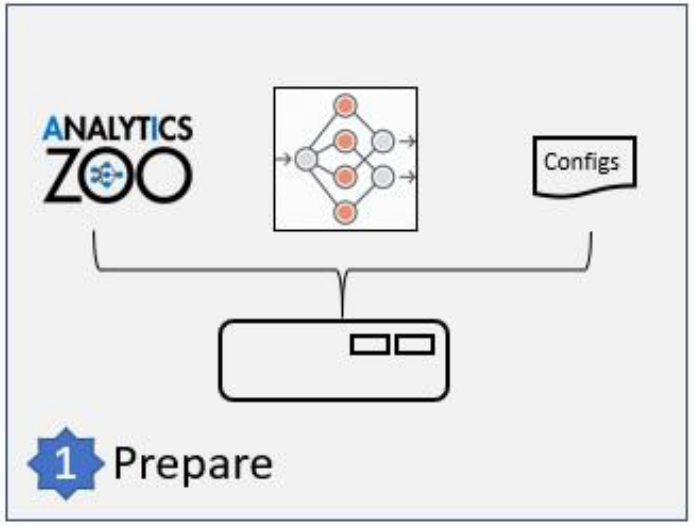
intel Software

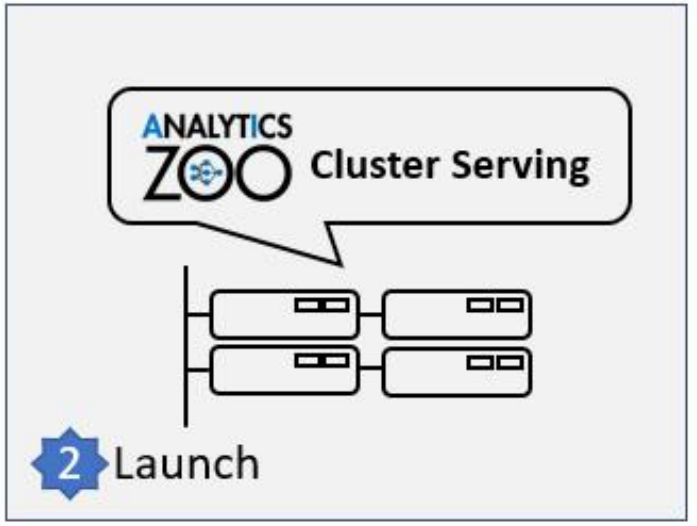
Analytics Zoo Cluster Serving

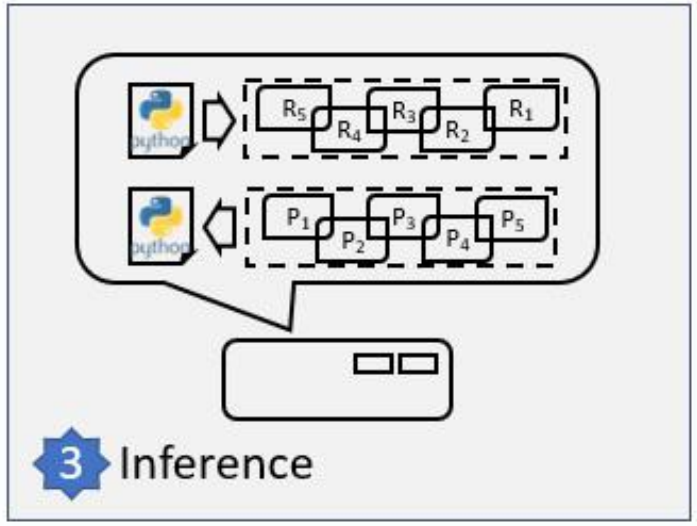


Cluster Serving Workflow Overview

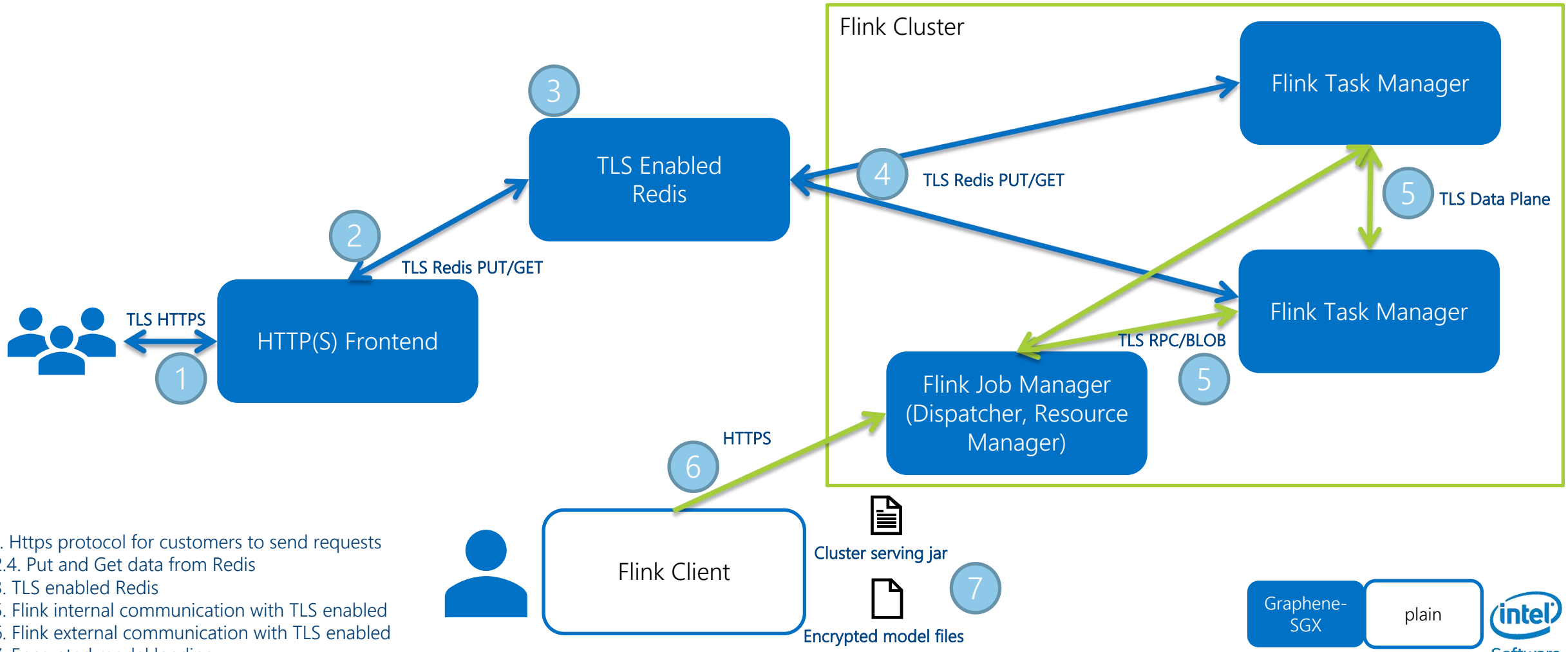
1. Install and prepare Cluster Serving environment on a local node
2. Launch the Cluster Serving service
3. Distributed, real-time (streaming) inference







Secured Cluster Serving on Graphene-SGX



1. Https protocol for customers to send requests
2.4. Put and Get data from Redis
3. TLS enabled Redis
5. Flink internal communication with TLS enabled
6. Flink external communication with TLS enabled
7. Encrypted model loading

Graphene-SGX | plain

intel Software

*Other names and brands may be claimed as the property of others.

Secured HTTP Frontend & Secured Redis

- Secured HTTP Frontend
  - TLS HTTPS enabled REST requests
  - TLS enabled Jedis GET/PUT to Redis
- TLS Enabled Redis
  - Rebuild Redis with BUILD_TLS=yes
  - HTTP Frontend TLS enabled Jedis GET/PUT to Redis
  - Flink Source/Sink TLS enabled Jedis GET/PUT to Redis

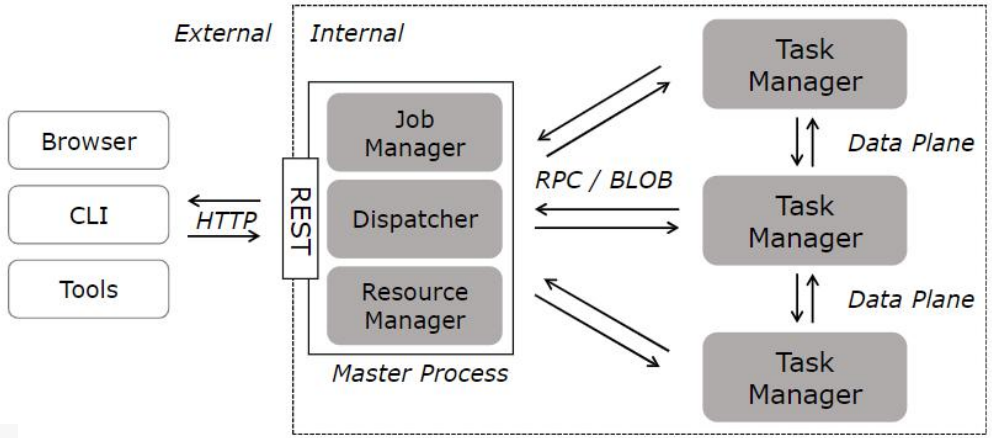
Secured Flink Cluster

- Flink SSL Setup and Internal and External Connectivity
  - Internal
    - nano flink-conf.yaml

```
security.ssl.internal.enabled: true
security.ssl.internal.keystore: /home/sgx/glorysdj/keys/keystore.pkcs12
security.ssl.internal.truststore: /home/sgx/glorysdj/keys/keystore.pkcs12
security.ssl.internal.keystore-password: XXXXXXXXr
security.ssl.internal.truststore-password: XXXXXXXX
security.ssl.internal.key-password: XXXXXXXXr
```

  - External

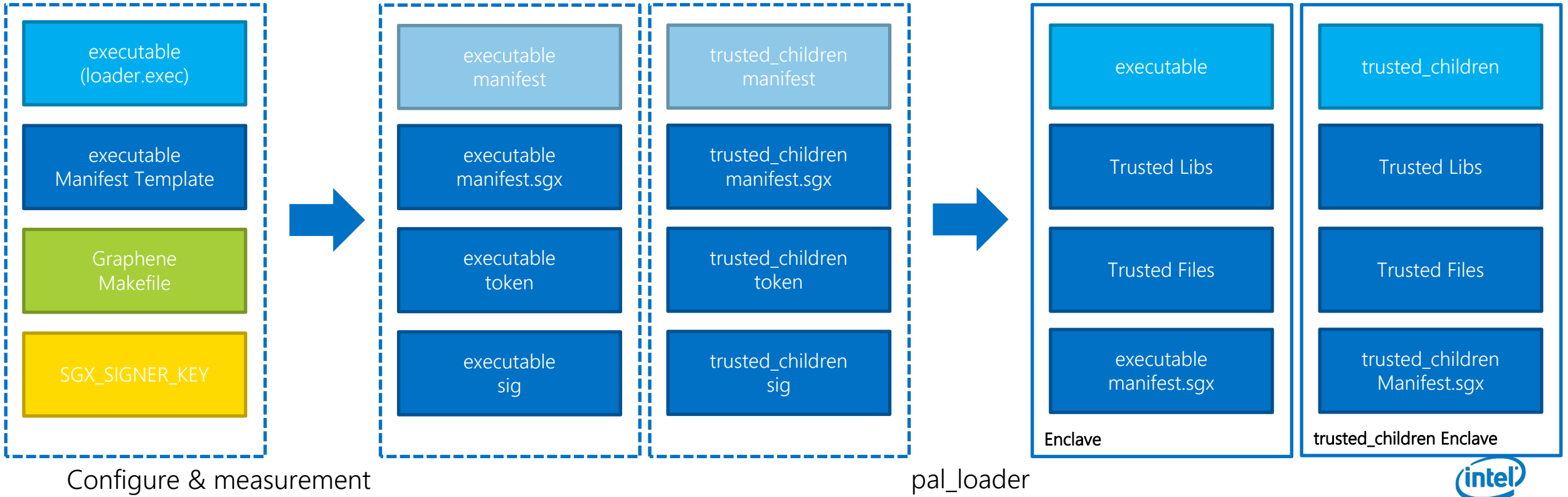
```
security.ssl.rest.enabled: true
security.ssl.rest.keystore: /path/to/flink/conf/rest.keystore
security.ssl.rest.truststore: /path/to/flink/conf/rest.truststore
security.ssl.rest.keystore-password: rest_keystore_password
security.ssl.rest.truststore-password: rest_truststore_password
security.ssl.rest.key-password: rest_keystore_password
```



Encrypted Model loading

- Encrypted Model loading
  - trait **EncryptSupportive**
    - def encryptWithAES256(content: String, secret: String, salt: String): String
    - def decryptWithAES256(content: String, secret: String, salt: String): String
    - def encryptFileWithAES256(filePath: String, secret: String, salt: String, outputFile: String, encoding: String = "UTF-8")
    - def decryptFileWithAES256(filePath: String, secret: String, salt: String): String
    - def decryptFileWithAES256(filePath: String, secret: String, salt: String, outputFile: String)
  - **InferenceModel**
    - def doLoadEncryptedOpenVINO(modelPath: String, weightPath: String, secret: String, salt: String, batchSize: Int = 0)
  - **ClusterServing**
    - secret/salt = jedis.hget(Conventions.MODEL_SECURED_KEY, Conventions.MODEL_SECURED_SECRET/...)
    - model.doLoadEncryptedOpenVINO(defPath, weightPath, secret, salt, coreNum)

Run Executable on Graphene-SGX



Run Analytics Zoo Secured Cluster Serving on Graphene-SGX

- Two Executables(loader.exec)
  - loader.exec = file:redis-server
  - loader.exec = file:/bin/bash
    - child enclaves: sgx.trusted_children.ls, sgx.trusted_children.cat, sgx.trusted_children.rm, …, sgx.trusted_children.java = file:/usr/bin/java
- Trusted Files: sgx.trusted_files.keys_keystore_*** = file:/home/sgx/keys/***
- Allowed Files: sgx.allowed_files.jvm = file:/usr/lib/jvm
- Enlarge enclave size: sgx.enclave_size = 32G
- Enlarge thread num: sgx.thread_num = 1024
- Run the commands
  - SGX=1 ./pal_loader redis-server …
  - SGX=1 ./pal_loader bash.manifest -c "……"

Thanks!