More virtualization and TEEs

CS503: Operating systems, Spring 2019

Pedro Fonseca
Department of Computer Science
Purdue University

Admin

- Ask for office hours if necessary (e.g., send email)
- Online students: also make sure you talk with TAs and me

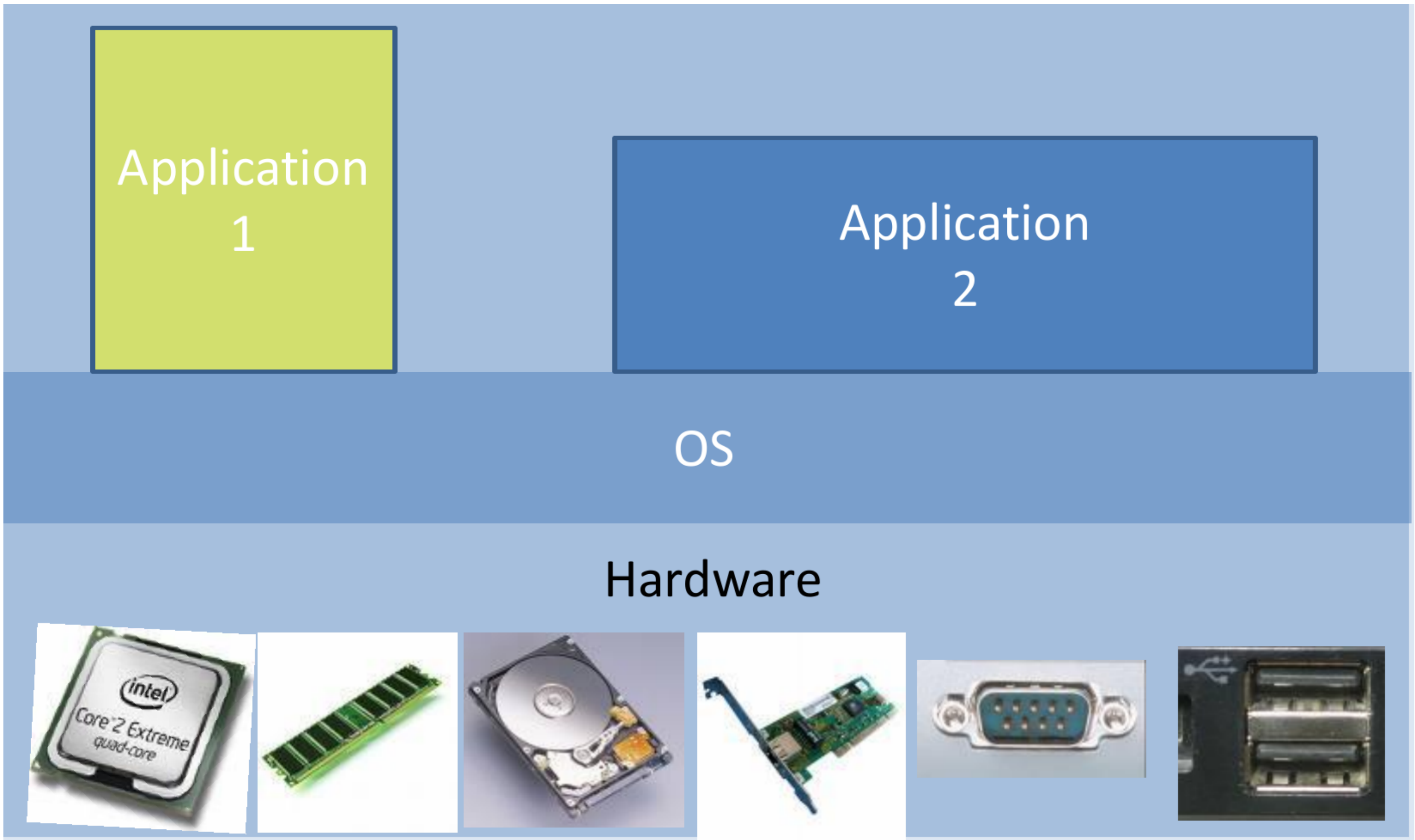
Next few lectures

- Advanced topics:
  - Virtualization / containers / SGX
  - Linux
  - Systems research
  - Operating systems + distributed systems
  - Review for exam

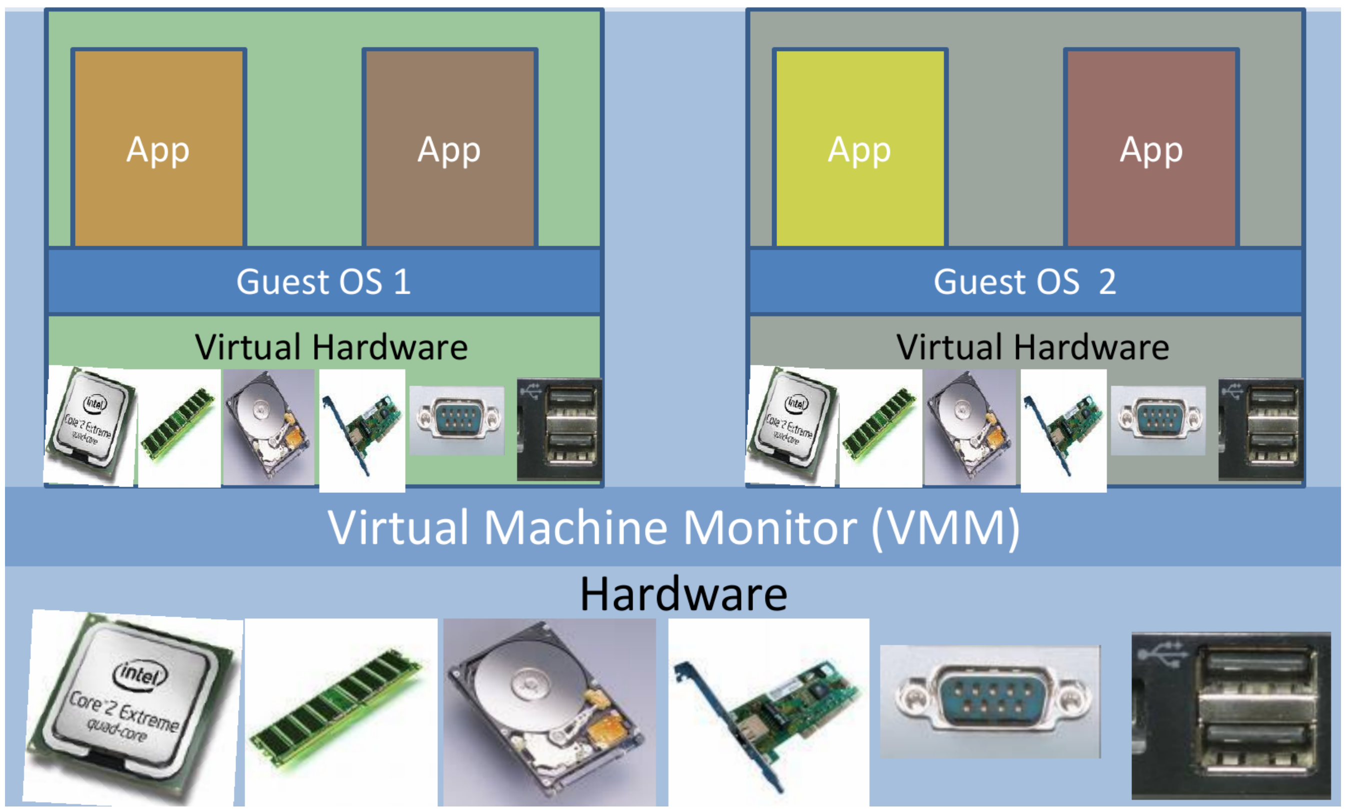
Last lecture

- VM-level virtualization
- Challenge in multiplexing
- Para-virtualization
- Advanced OS features
  - Checkpoint and restart
  - Migration

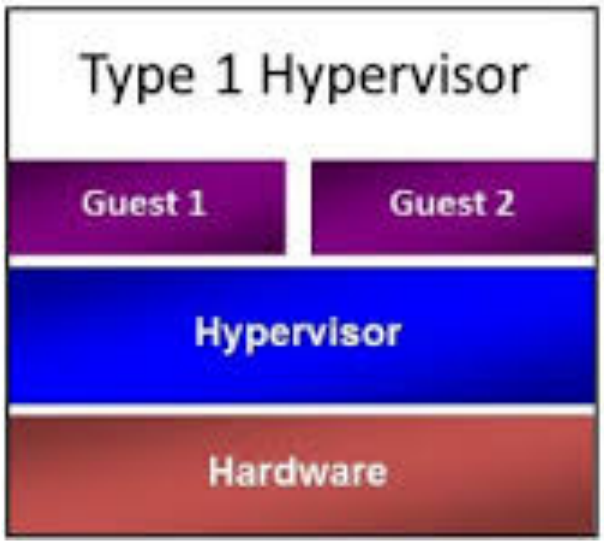
Recall: Traditional

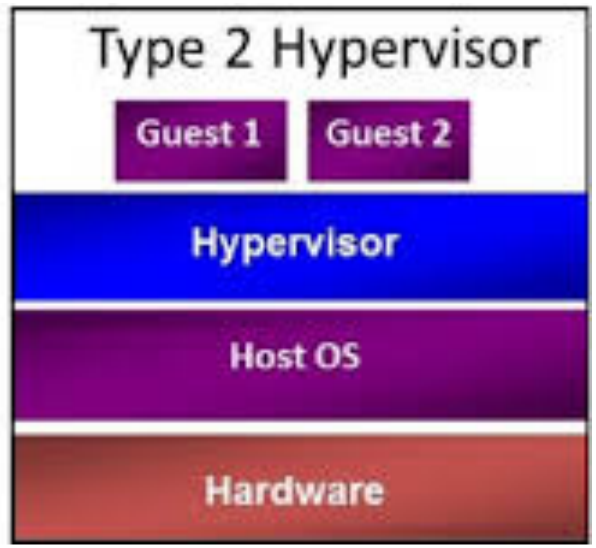


Recall: VM virtualization



Type 1 vs. type 2 hypervisors





e.g., Xen

e.g., VirtualBox, QEMU/KVM

What are the advantages of each?

Recall: Advanced topics and discussion

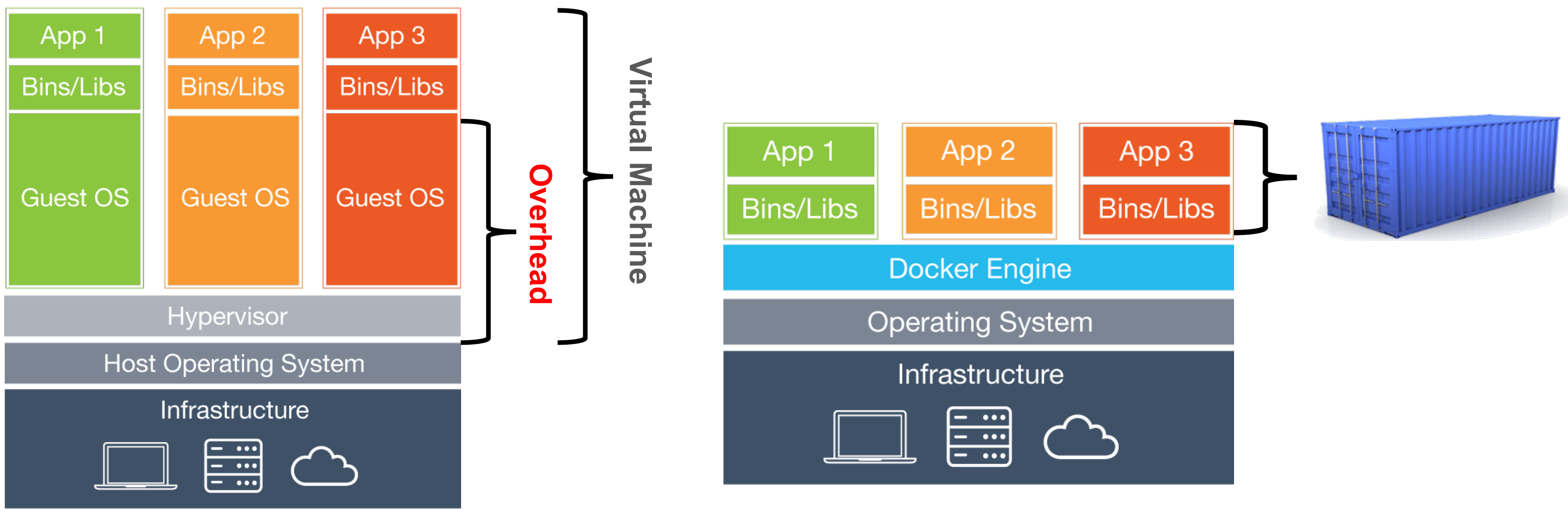
- Scheduling multiple VMs
  - How does this affect the VMM?
  - Impact of caches
- Suspend and resume
- Migration of VMs
  - What are the uses?
  - What are the challenges?

Process-level virtualization (containers)

Process-level virtualization

- Also known as *OS-level virtualization*:
  - Used to implement the *container* abstraction
- Run application in an isolated environment

VMs vs. containers



- Containers are generally “lighter” than VMs

	Container	Virtual Machine	Bare-Metal x86 Server
Underlying Platform	OS on Virtual Machine or Bare-Metal x86 Server	Hypervisor on Bare-Metal x86 Server	N/A
Performance: Speed and Consistency	Average	Average	Fastest
Provisioning Time	Seconds	Minutes	Hours
Tenant Isolation Enforcement	OS Kernel	Hypervisor	Physical
Ideal Application Types	Mode 2	Mode 1 or Mode 2	Mode 1 or Mode 2
Configuration and Reconfiguration Flexibility	Highest	Medium	Lowest
Host Consolidation Density	Maximum	Average	None
Application Portability	Application Packaging/ Manifest*	VM Image, VM Migration Tools	Backup and Restore, ISO Images
Granularity	Extremely Small	Average	Largest

*While application portability is somewhat easier in container environments that are leveraging a container management and orchestration solution, portability should not assumed to be universal — differences in the underlying host OS below the containers could still present some interoperability challenges.

Source: Gartner (September 2015)

*** Take this table with a grain of salt**

Containers

- Recent trend to develop and deploy applications:
  - Ensures the same environment
  - Sandbox application / project
  - Automatically deals with dependencies
- Containers usually rely on
  - OS services to implement isolation
  - A library that implement a common environment
  - Tools to manage containers

Using containers with Docker

- Write a “dockerfile”:

```
FROM ubuntu:18.04
COPY . /app
RUN make /app
CMD python /app/app.py
```

- Build an “image” (layered read-only FS)
- Deploy the image on a machine
- Other container management systems: rkt, LXC

Q: And how to use a VM?

https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

Discussion

- What are the (minimum) OS services required to implement containers?
- Container orchestration (e.g., Kubernetes)
- When are VMs better?
- Are there security differences
- How to address the network isolation? Would it be useful?

Trusted execution environments

Security issues with the cloud

- Future of clouds: all your data will be in clouds
  - Email
  - Location-based service
  - Bitcoin
  - Password management service
  - DNA matching service
- Q. Should you trust clouds?
  - Should you trust Amazon when using Amazon EC2?
  - Should you trust Google when using gmail?

Trusted execution environments

- Intel SGX, ARM TrustZones
- Implement the secure **enclave** abstraction
- Provide several properties to the code running inside the enclave

Properties: confidentiality

- An adversary outside of the enclave cannot inspect the state of execution inside the enclave
- Even if the OS is compromised or execute a malicious application on the machine

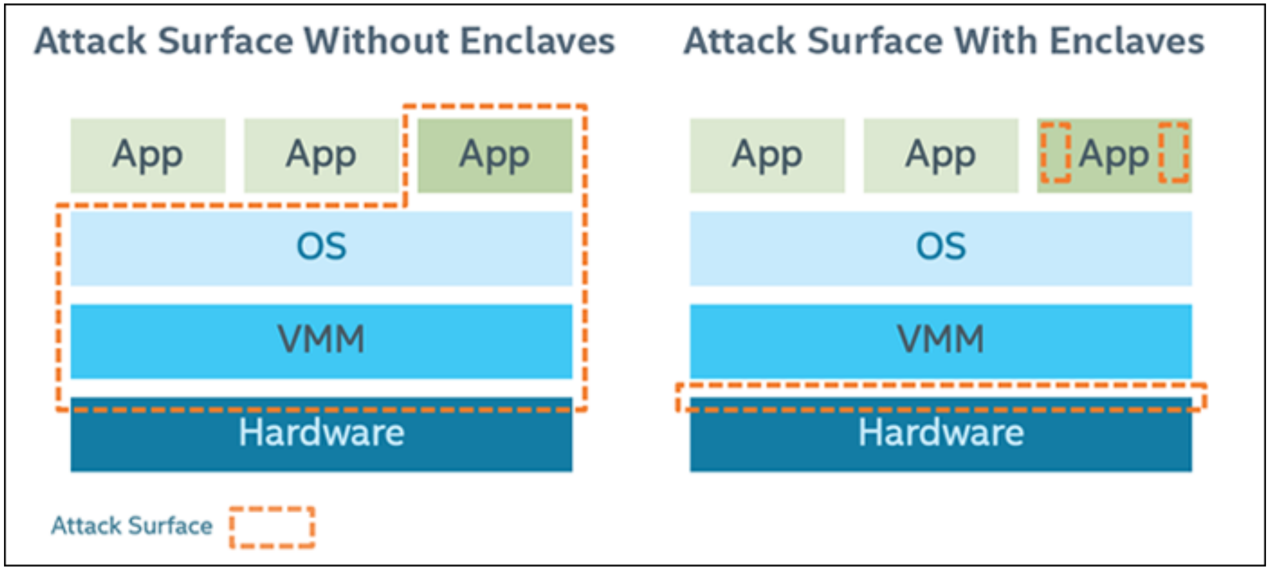
Properties: integrity

- Guarantees that an attacker outside of the enclave cannot affect the result of computation inside the enclave:
  - Except from supplying inputs through the defined interface
- Ensures correctness of the computation running inside the enclave even if the OS

Properties: attestation

- Provides an (unforgeable) proof that enables a remote party to verify what has run inside the enclave
- Even if they don’t have physical access to the machine
- Relies on cryptography (and requires trusting the HW)

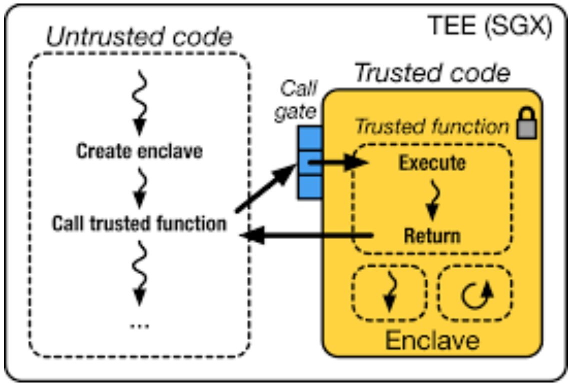
SGX attack surface



- Requires special hardware
- Hardware-support can also prevent physical attacks by reading the memory bus

Q: SGX + hypervisor makes sense?

Secure computation with SGX enclaves



Discussion topics

- Hows does SGX compare with user/kernel protection?
- SGX + hypervisors makes sense
- What cryptography is necessary? Where are keys stored?
- Any important properties not guaranteed by SGX?
- Practical security
- Mathematical alternatives to HW-based solutions?

Summary

- Hypervisors
  - Type 1 vs. type 2
  - Advantages
- OS-level / process-level virtualization
  - Container abstraction
  - Light virtualization
  - Performance, isolation, security
- Trusted execution environments