Everywhere All at Once: Co-Location Attacks on Public Cloud FaaS

Zirui Neil Zhao, Adam Morrison, Christopher W. Fletcher, Josep Torrellas

University of Illinois Tel Aviv University

To appear at ASPLOS'24

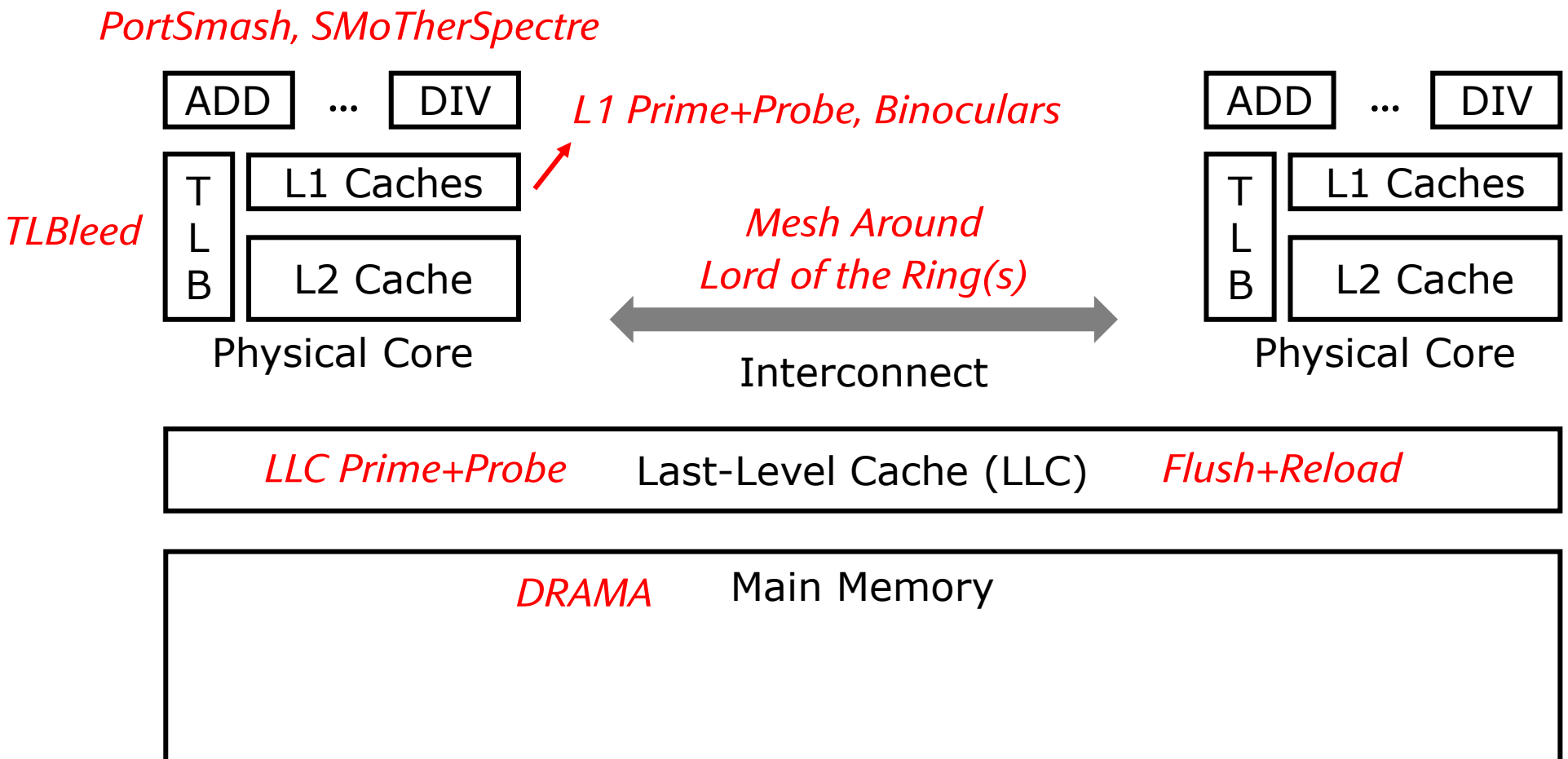
ILLINOIS

TEL AVIV UNIVERSITY

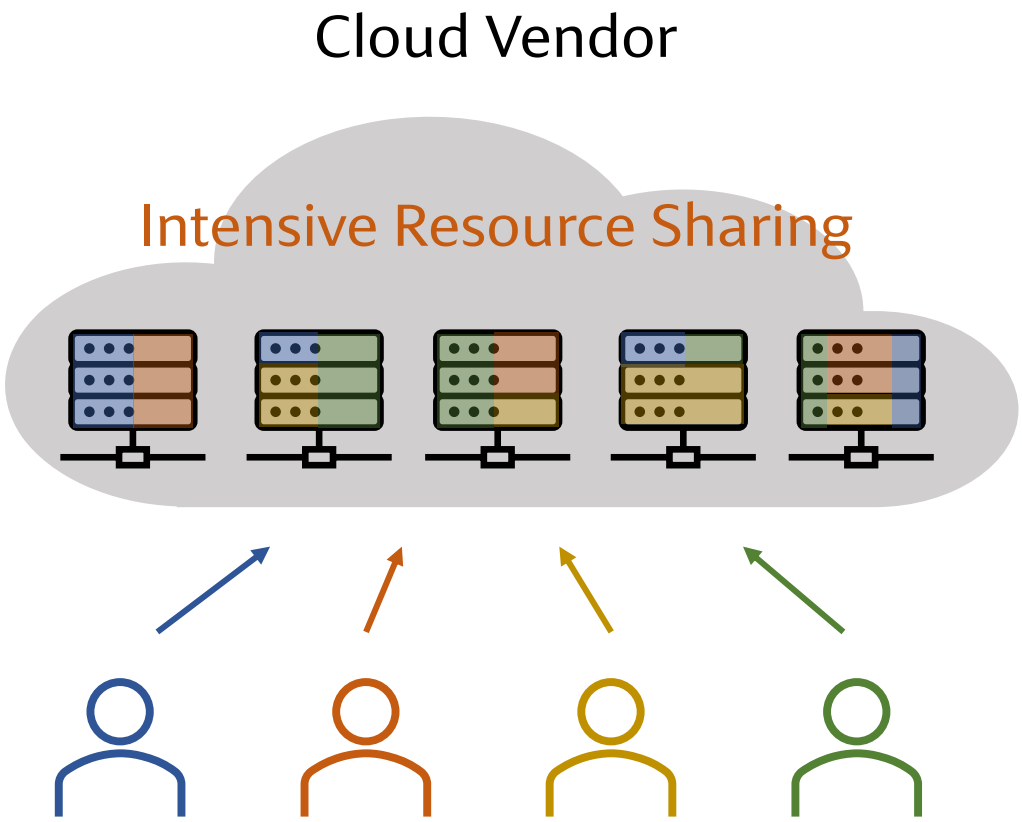
Ad *Will be on the job market this fall

Microarchitectural Side-Channel Attacks

Side channels exploit **shared resources** between attackers and victims

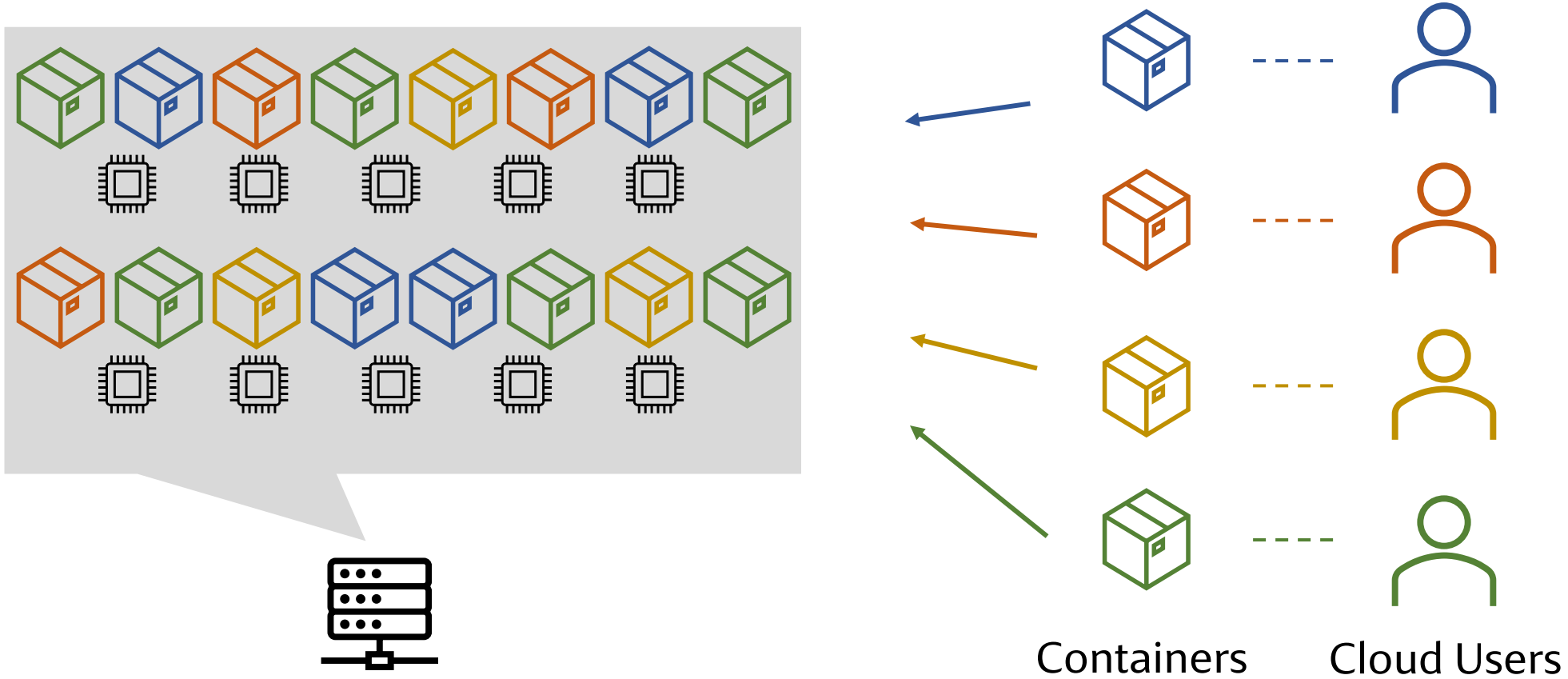


Cloud Computing: A Prime Target for Side Channels

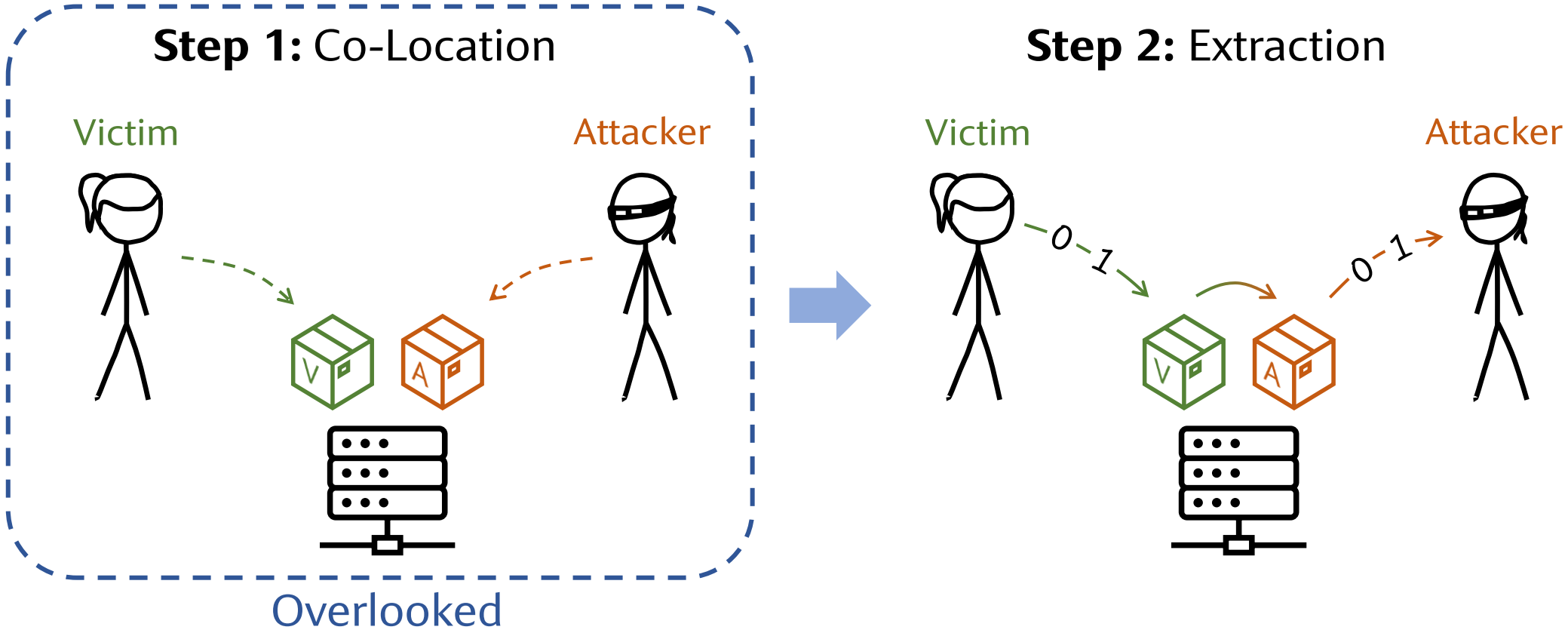


Emerging Cloud Computing Has Even More Sharing

Example: Function-as-a-Service (FaaS)



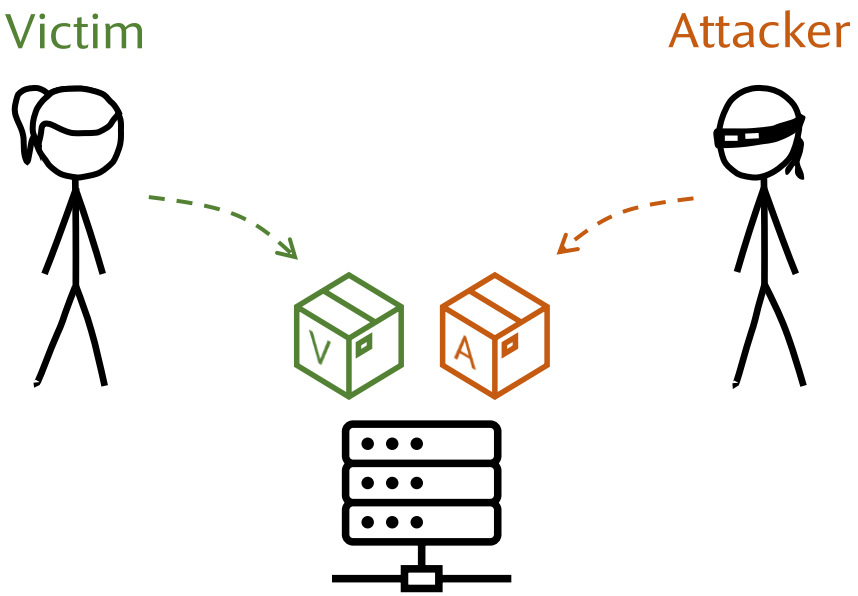
Steps of Side-Channel Attacks in the Wild



*Characters are based on https://xkcd.com/2176 and https://xkcd.com/1808/ (under a CC Attribution-NonCommercial 2.5 License)

Goal of This Work

Step 1: Co-Location

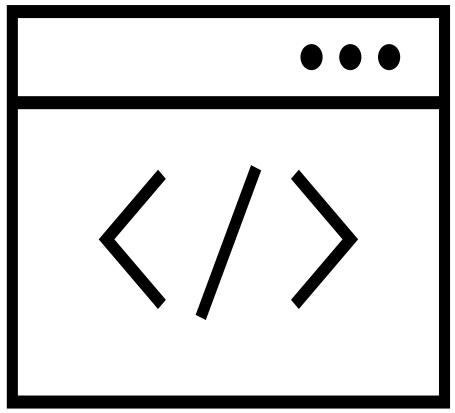


Goal: study how to achieve co-location in modern public clouds

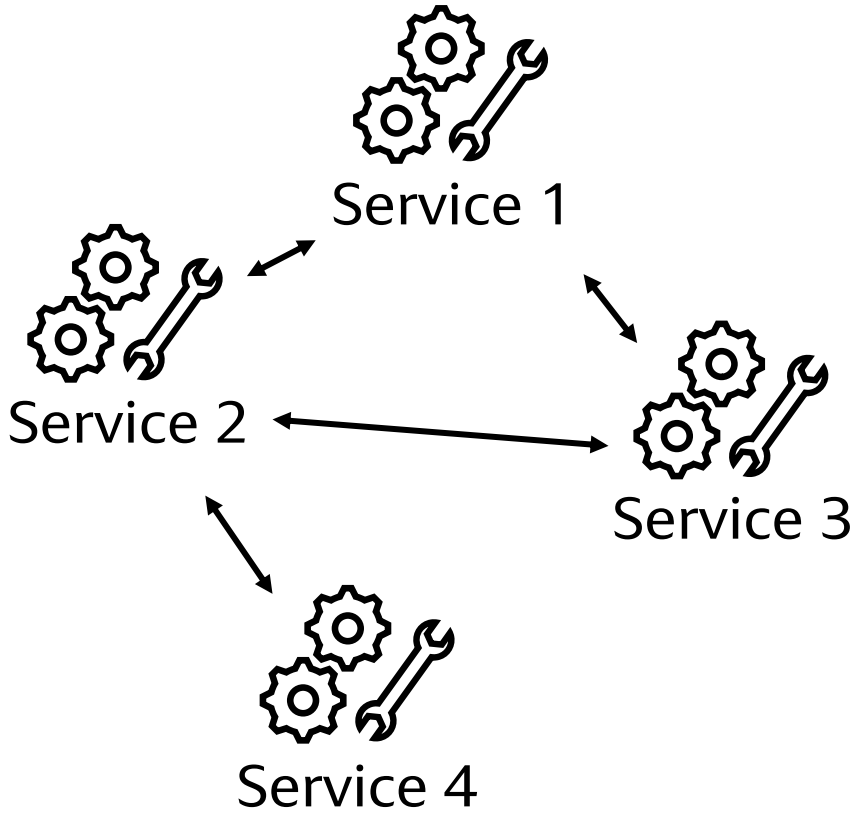
Target: Google Cloud Run

A production Function-as-a-Service platform from Google Cloud

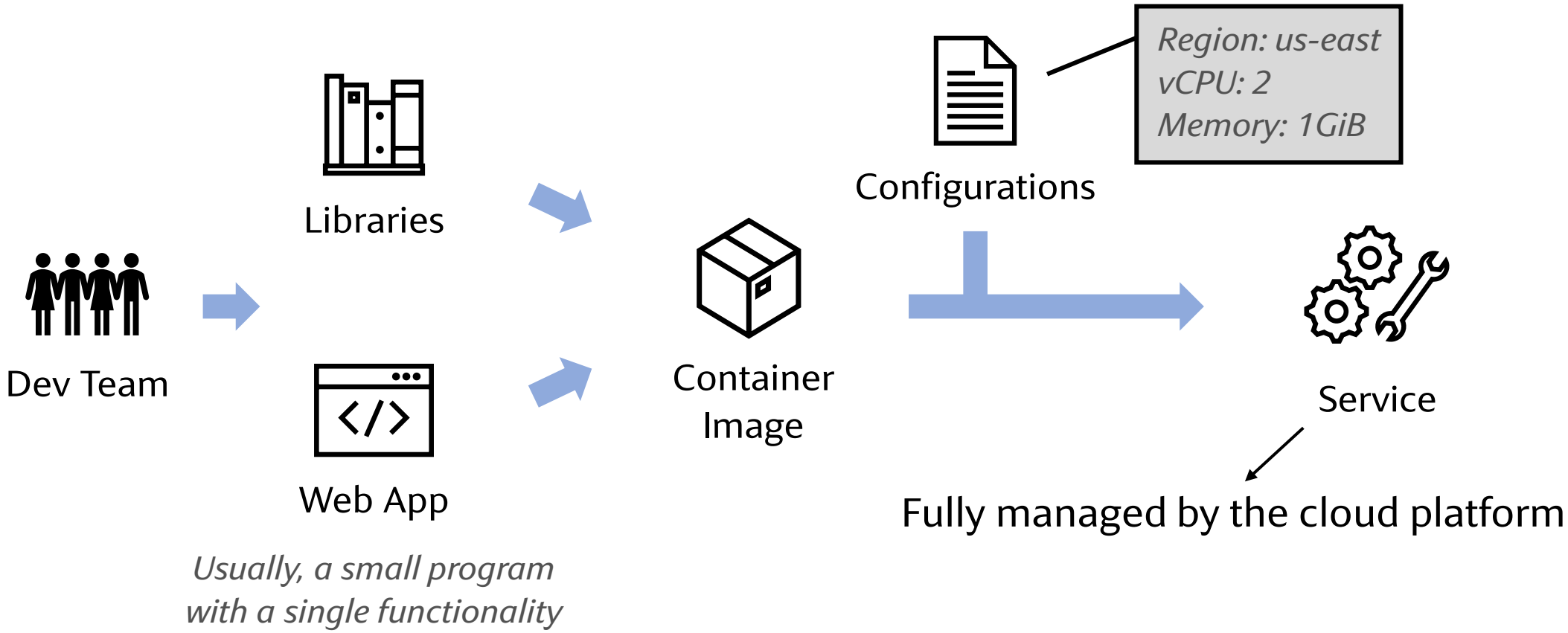
Background: Function-as-a-Service (FaaS)



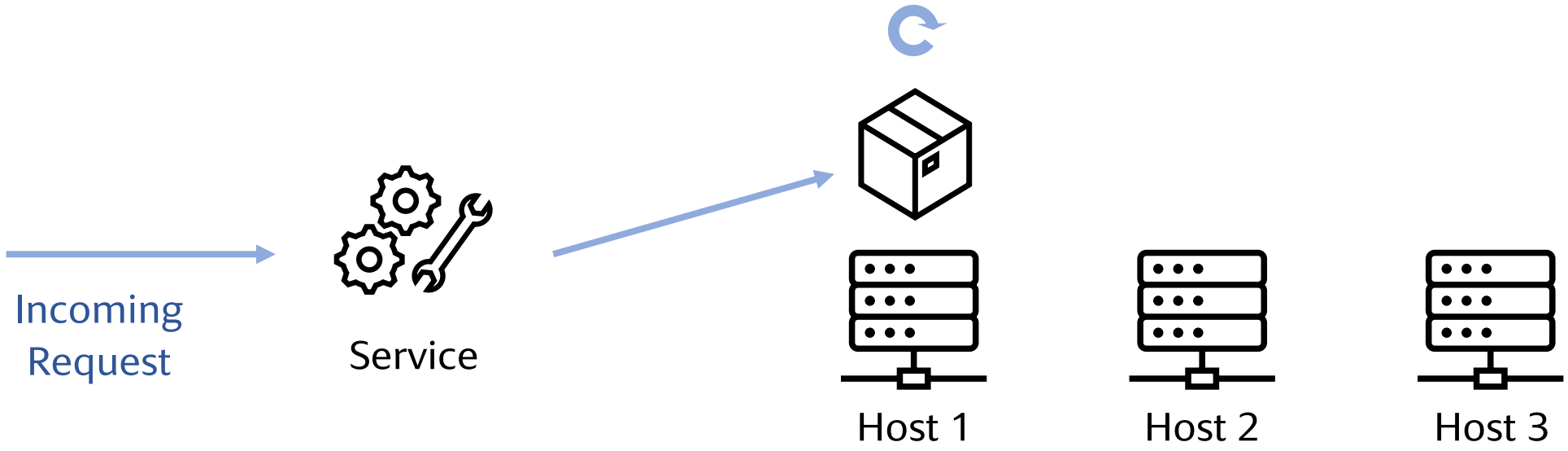
Monolithic App



Background: Function-as-a-Service (FaaS)

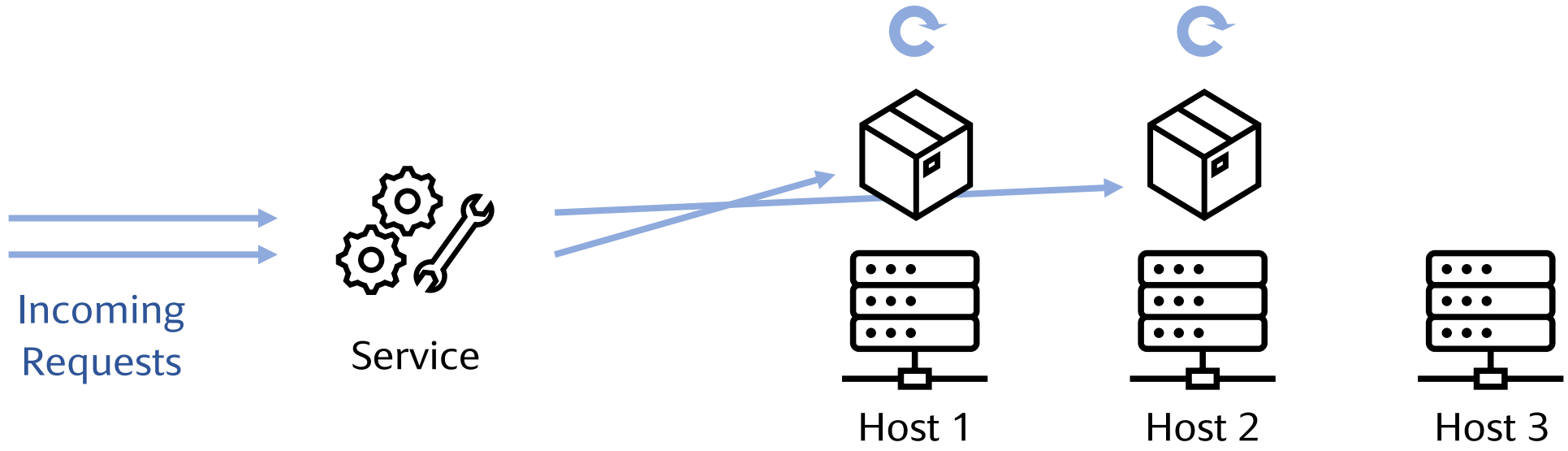


Background: Request Handling and Auto-Scaling



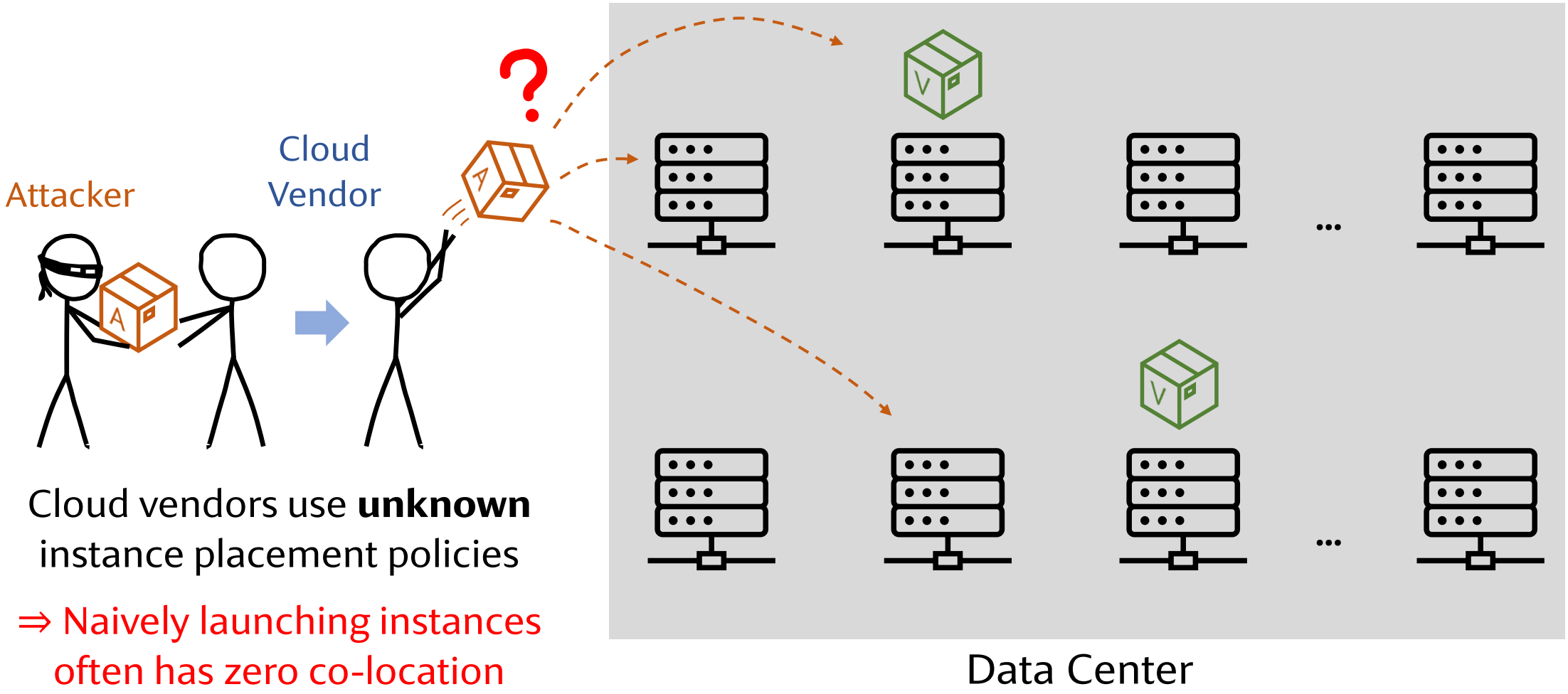
The instance launching process is fully managed by the cloud platform

Background: Request Handling and Auto-Scaling



Cloud platform launches more instances to handle traffic increases

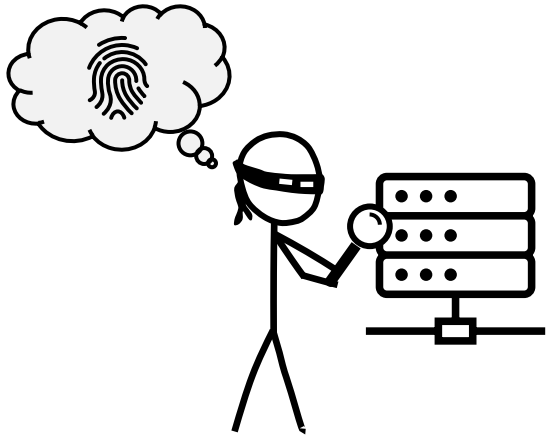
Fog of War: Container Instance Placement



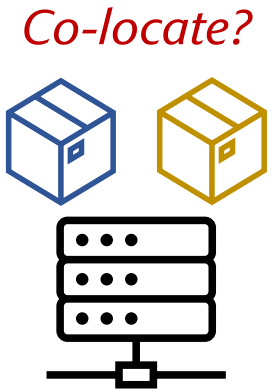
Cloud vendors use **unknown** instance placement policies

⇒ Naively launching instances often has zero co-location

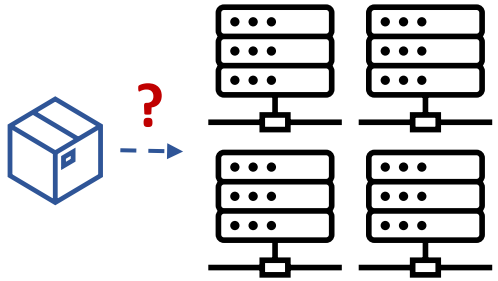
Main Contributions & Highlighted Results



1. Accurate
Host Fingerprinting



2. Inexpensive
Co-Location Test



3. Exploitable
Placement Behavior

100% probability of co-locating with at least one victim instance

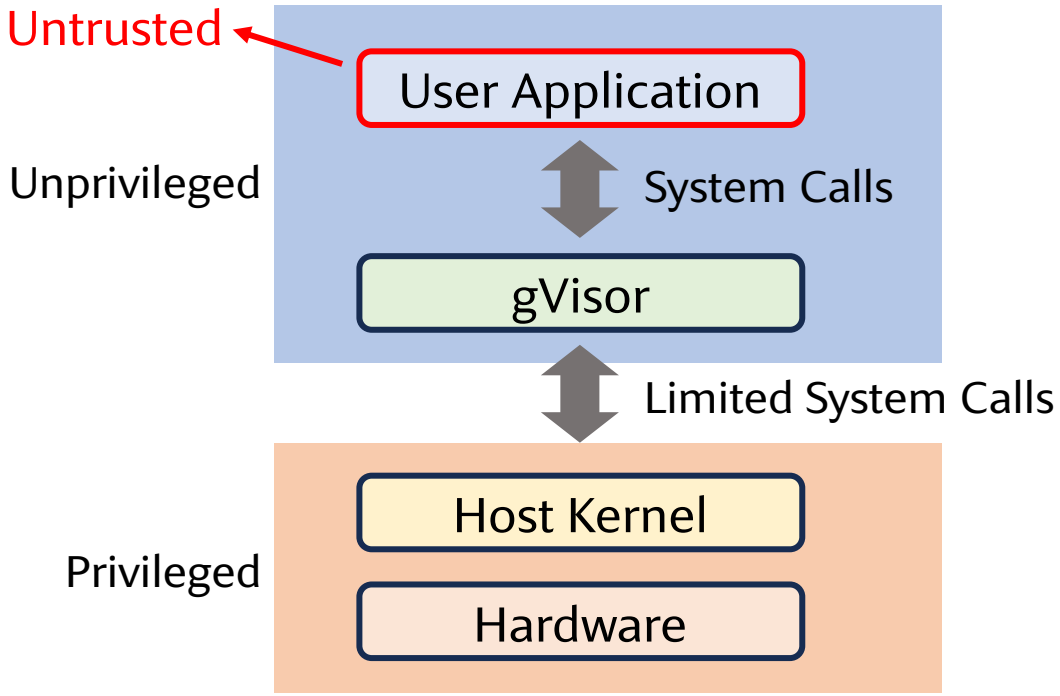
Co-locate with 61%-100% of victim instances in three major data centers

Threat Model

- An unprivileged attacker
- After co-location is achieved, the attacker can invoke the victim and exfiltrate information
- The attacker can execute arbitrary program inside the container

Securing the Cloud: Sandboxing Technologies

Linux Containers & gVisor

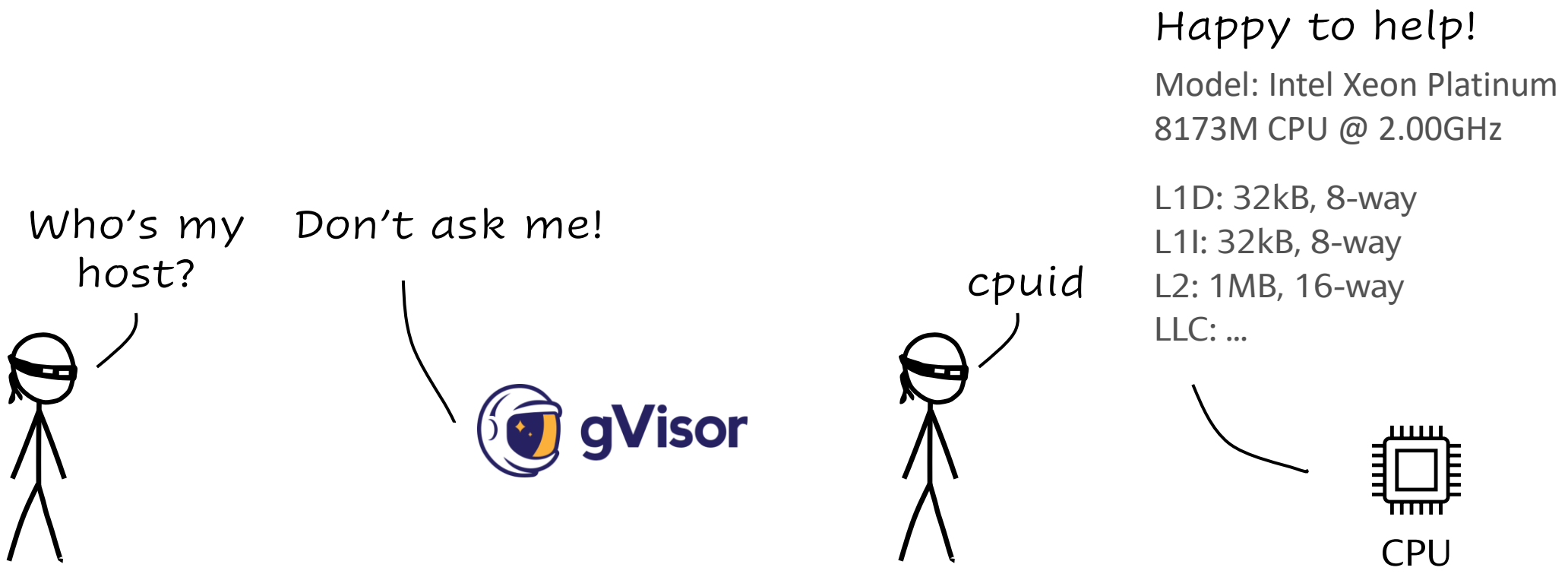


Non-virtualized, lightweight
Default environment for **web services**

gVisor Hides Sensitive Host Information

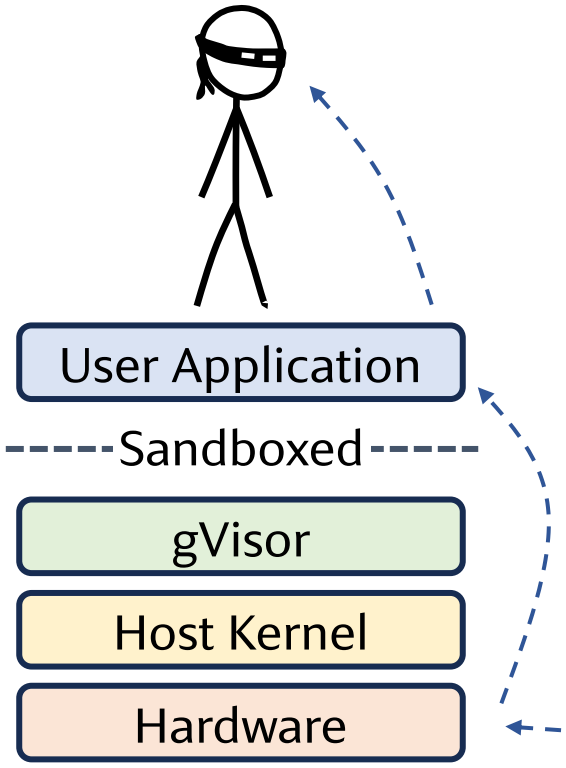
```
Attacker: CPU Model?
gVisor: unknown

Attacker: Boot log?
gVisor:
  Starting gVisor...
  Granting licence to kill(2)...
  Recruiting cron-ies...
  Creating process schedule...
  Checking naughty and nice process list...
  Gathering forks...
  Rewriting operating system in Javascript...
  Searching for needles in stacks...
  …
```

Physical Host Fingerprinting: Talk with the CPU



Insight: attacker can bypass software countermeasures by directly interacting with the shared underlying hardware

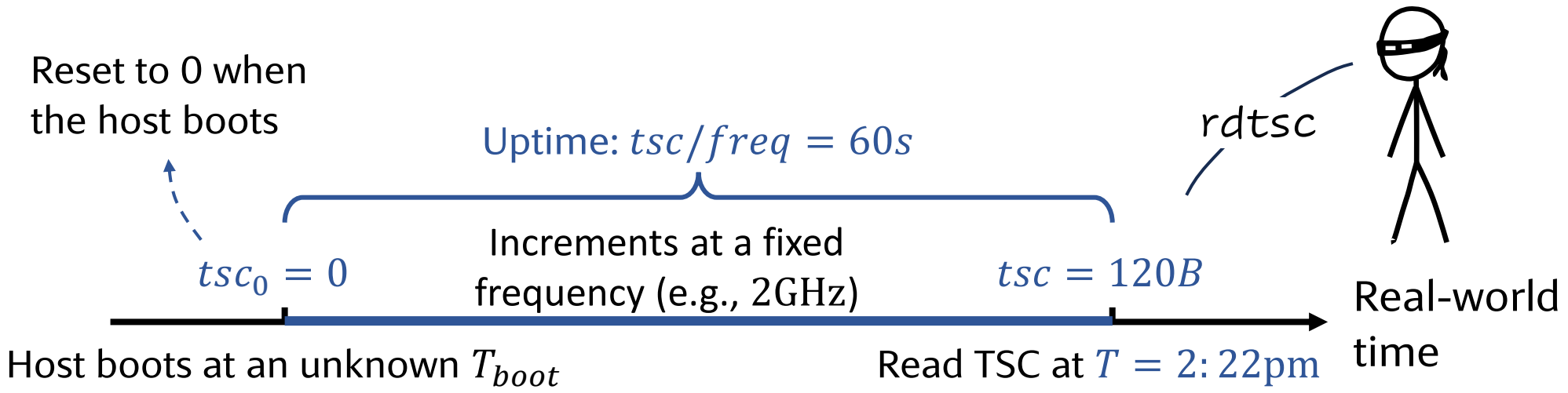
My Little Timestamp Counter (TSC)



The host's TSC is accessible through an unprivileged x86 instruction **`rdtsc`**

A high-resolution timer (frequency ~ a few GHz)

Deriving Host's Boot Time Using TSC

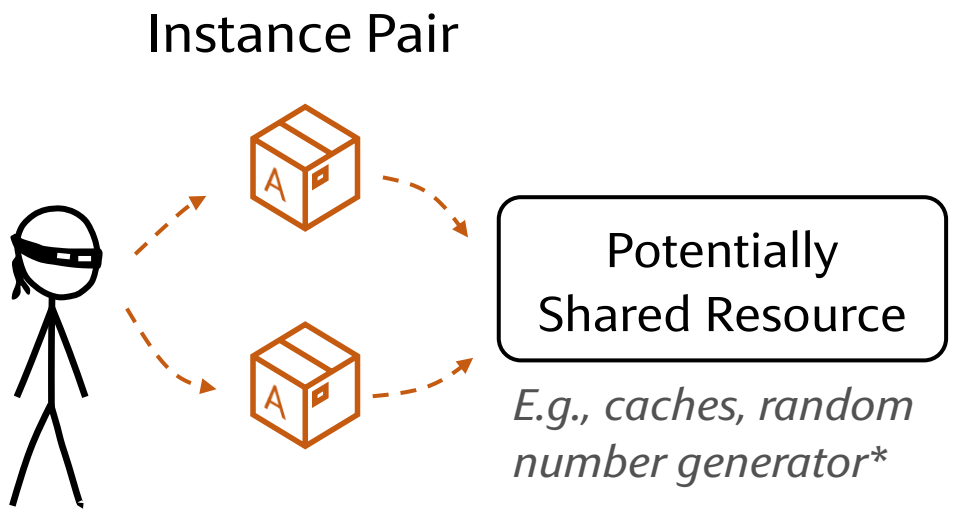


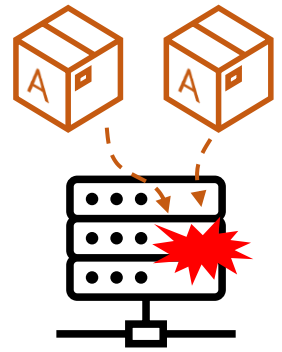
$$\Rightarrow T_{boot} = T - uptime = 2{:}21\text{pm}$$

Hypothesis: the boot time is likely unique for each physical host

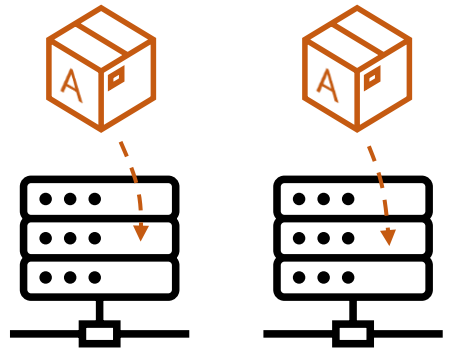
Host Fingerprint: $\mathbb{F} = T_{boot}$

Verifying Co-Location





Contention!
⇒ Co-Located

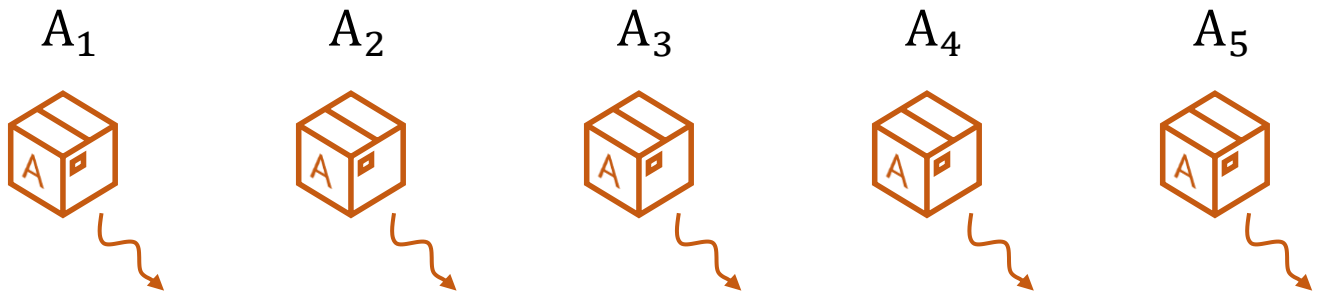


No Contention
⇒ Not Co-Located

Scalability Issue: it requires $O(N^2)$ pairwise tests to verify N containers

* Evtyushkin et al. Covert Channels through Random Number Generator: Mechanisms, Capacity Estimation and Mitigations

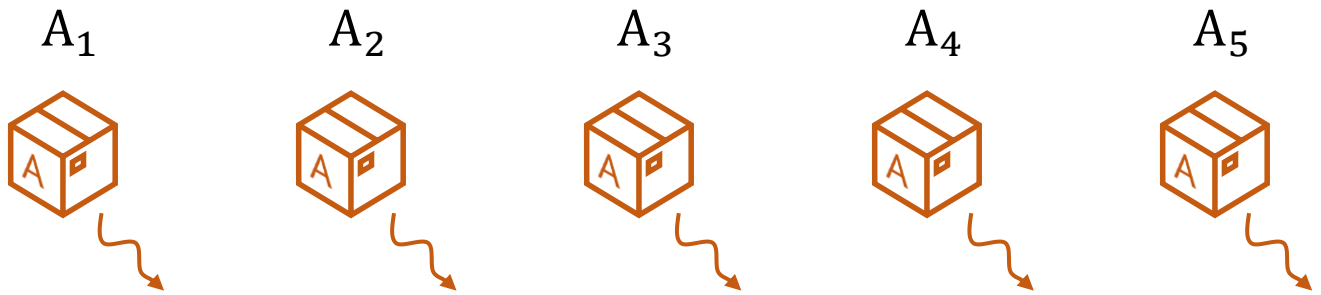
Go Beyond Pairwise Testing – Batch Testing



A_1, A_2, A_3 do not co-locate with any other instance
(i.e., they are *single instances*)

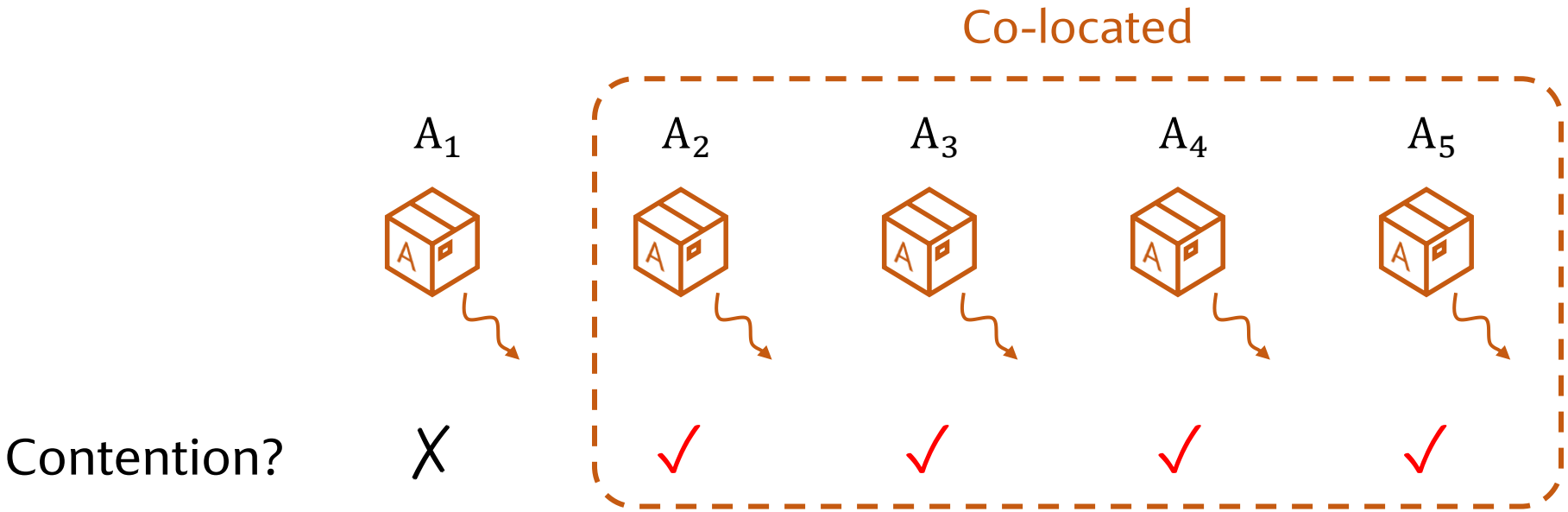
A_4 and A_5 are co-located

Go Beyond Pairwise Testing – Batch Testing



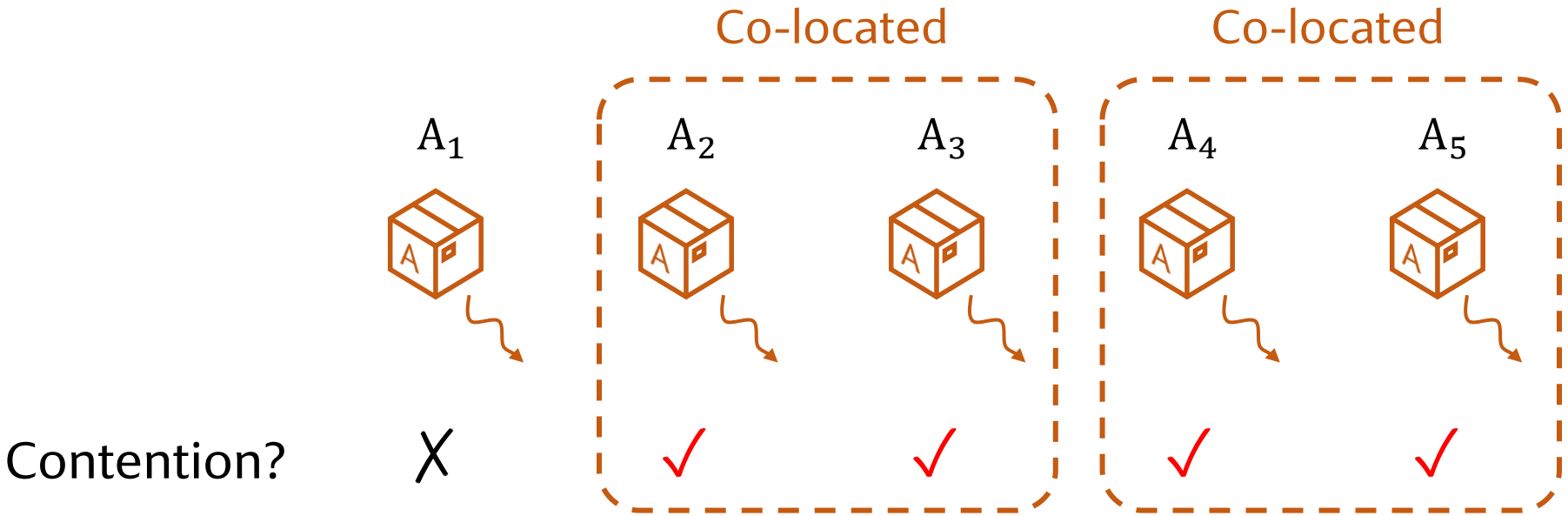
A_3, A_4, A_5 are co-located

Go Beyond Pairwise Testing – Batch Testing



A_2, A_3, A_4, A_5 are co-located? Not sure!

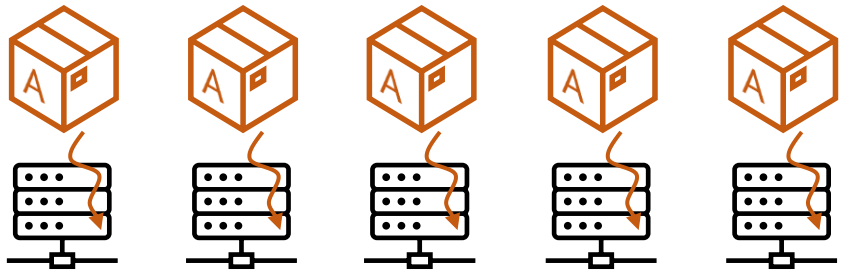
Go Beyond Pairwise Testing – Batch Testing



A_2, A_3, A_4, A_5 are co-located? Not sure!

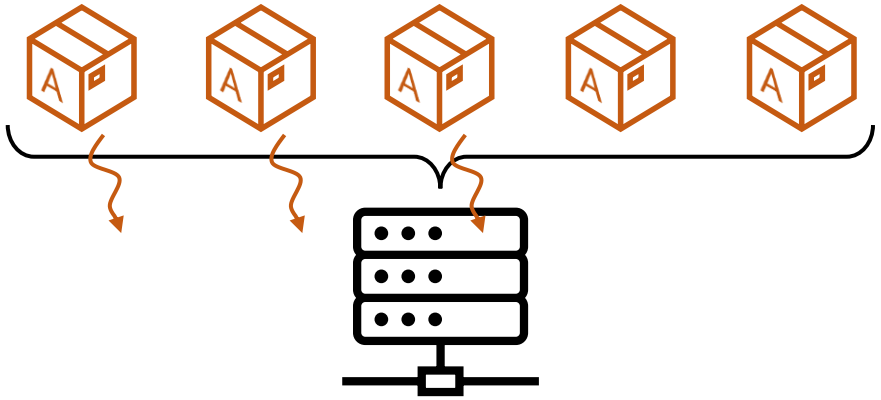
Batch Testing Strategies

Truly not co-located



Batch test all instances at once

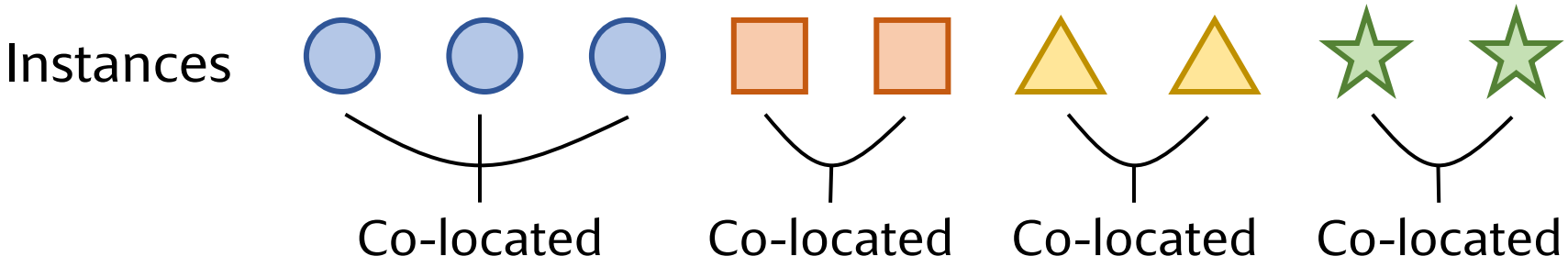
Truly co-located



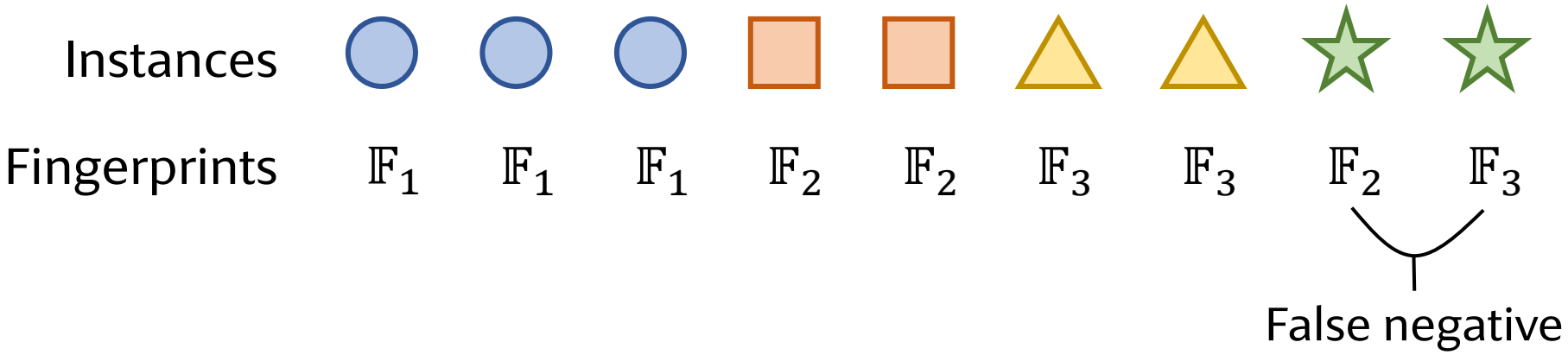
Batch test 3 instances at once

If fingerprints are accurate, they can provide hints on which instances are likely co-located

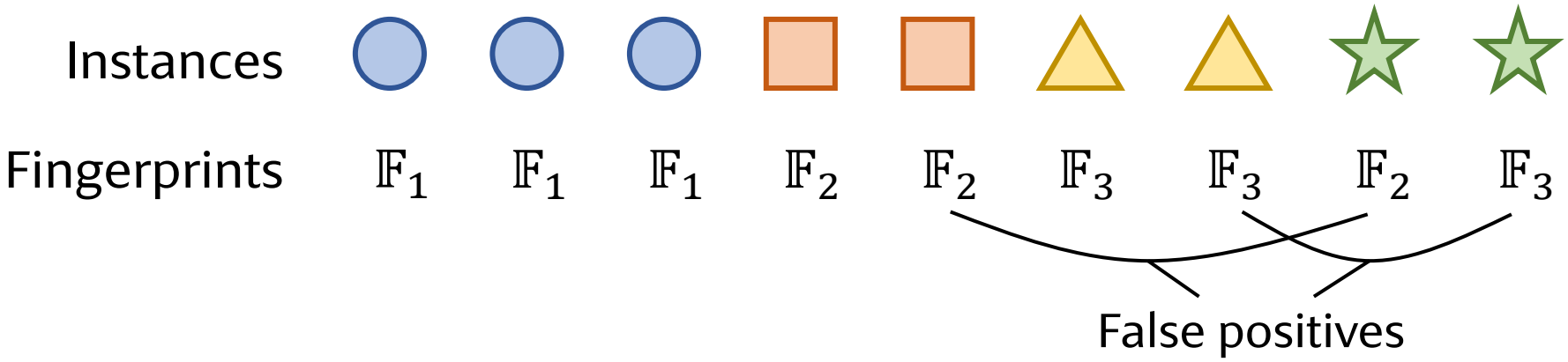
Fingerprint-Assisted Co-Location Verification



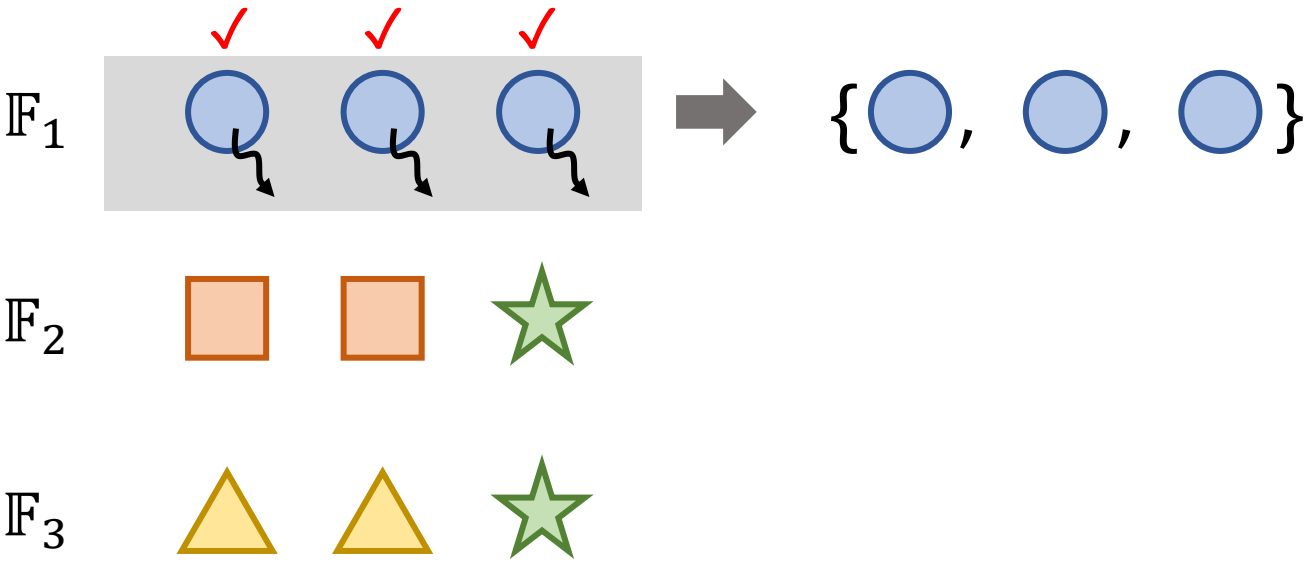
Fingerprint-Assisted Co-Location Verification



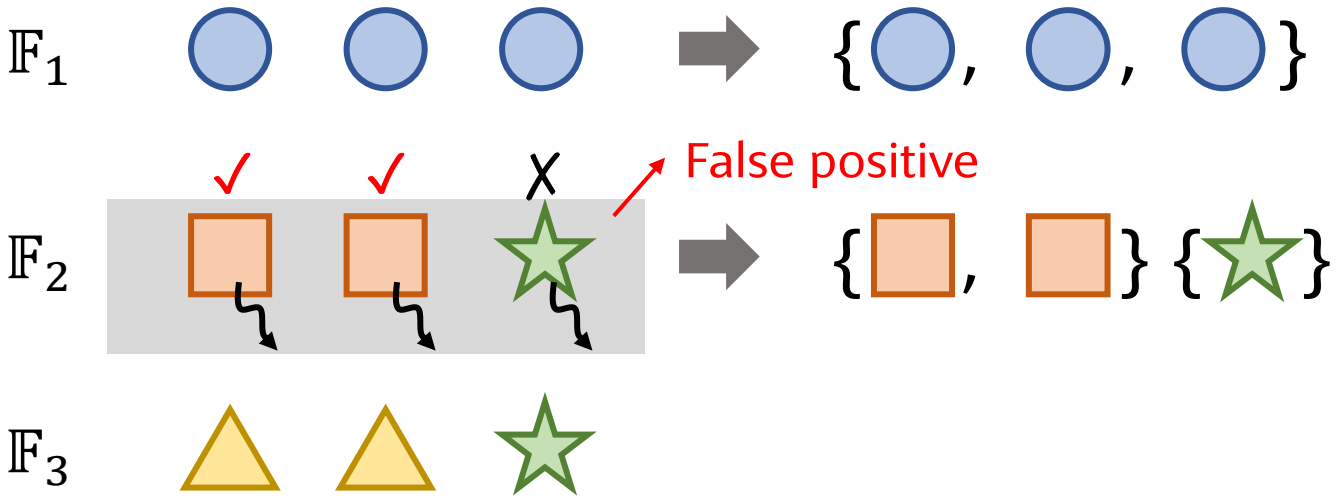
Fingerprint-Assisted Co-Location Verification



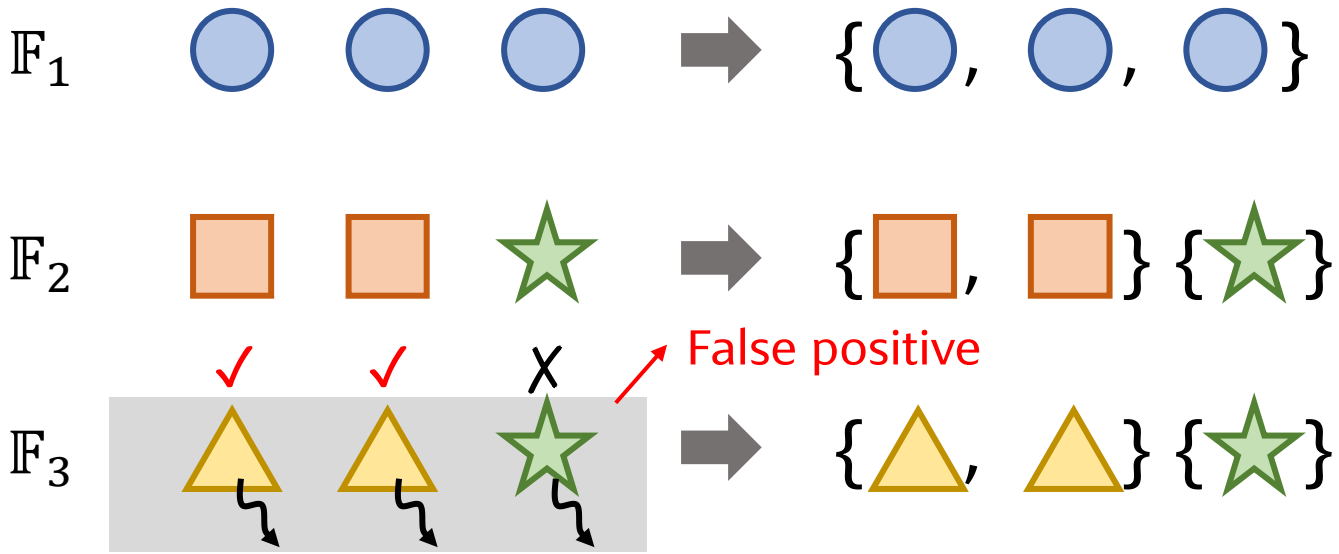
Fingerprint-Assisted Co-Location Verification



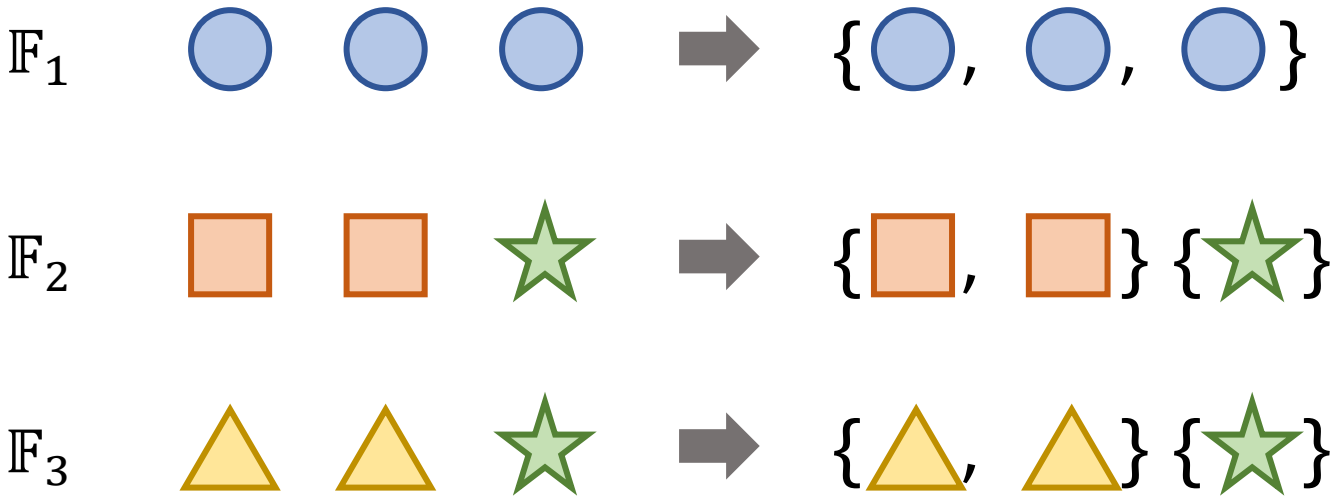
Fingerprint-Assisted Co-Location Verification



Fingerprint-Assisted Co-Location Verification

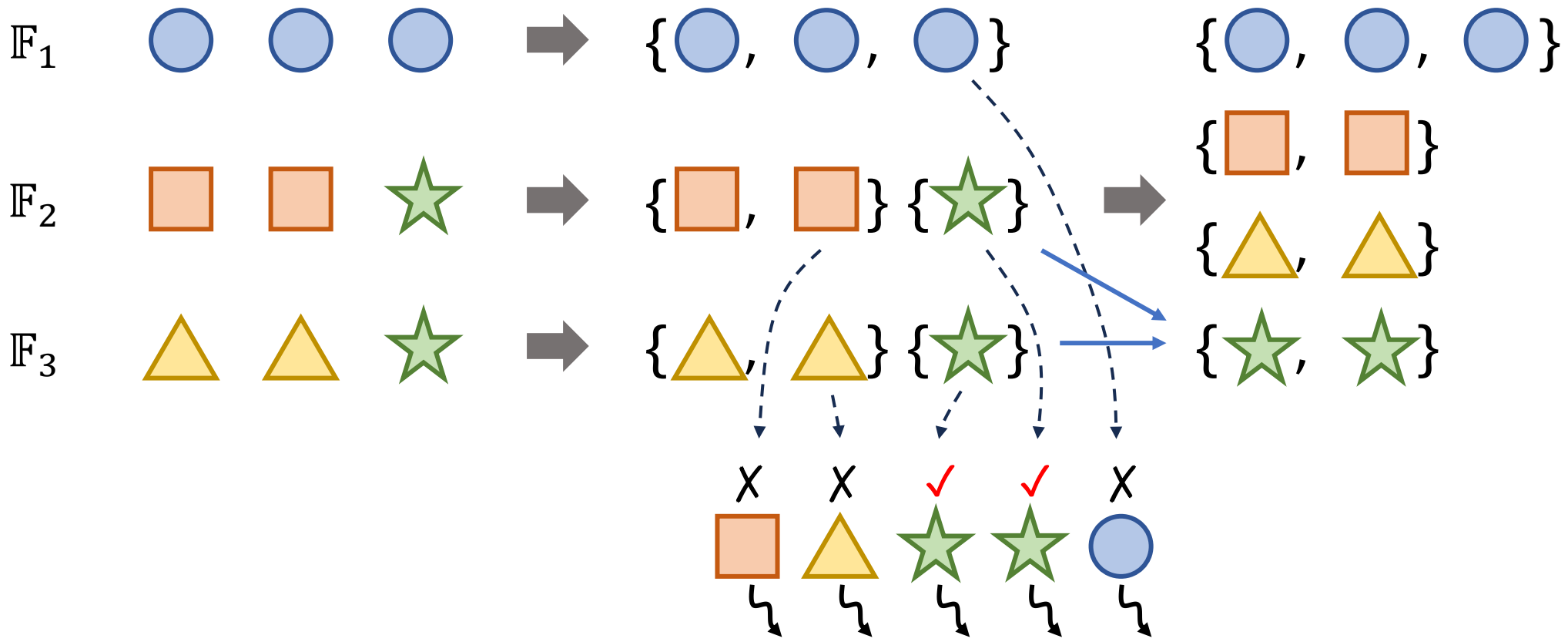


Fingerprint-Assisted Co-Location Verification

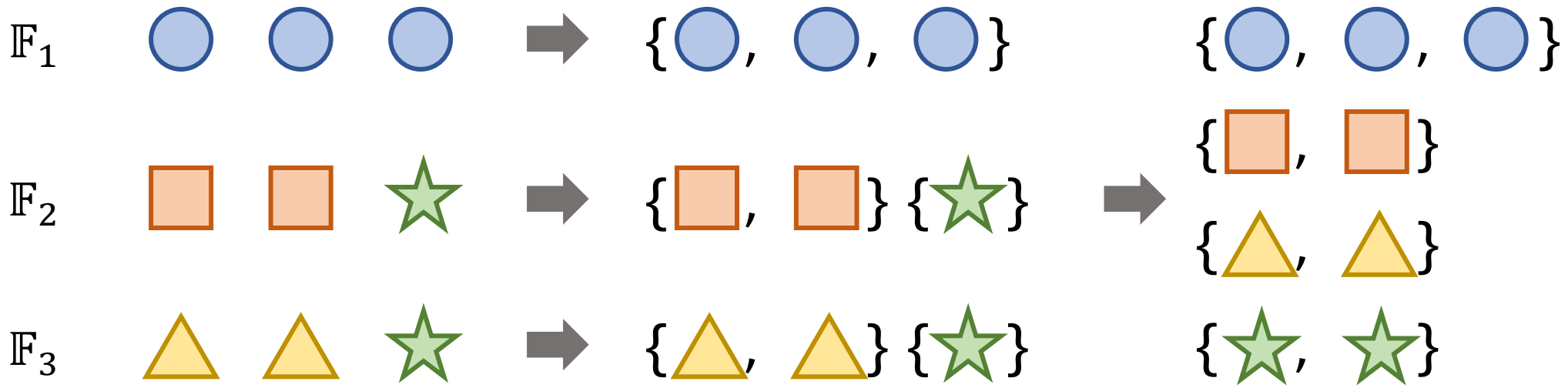


All false positives are identified, proceed to find false negatives

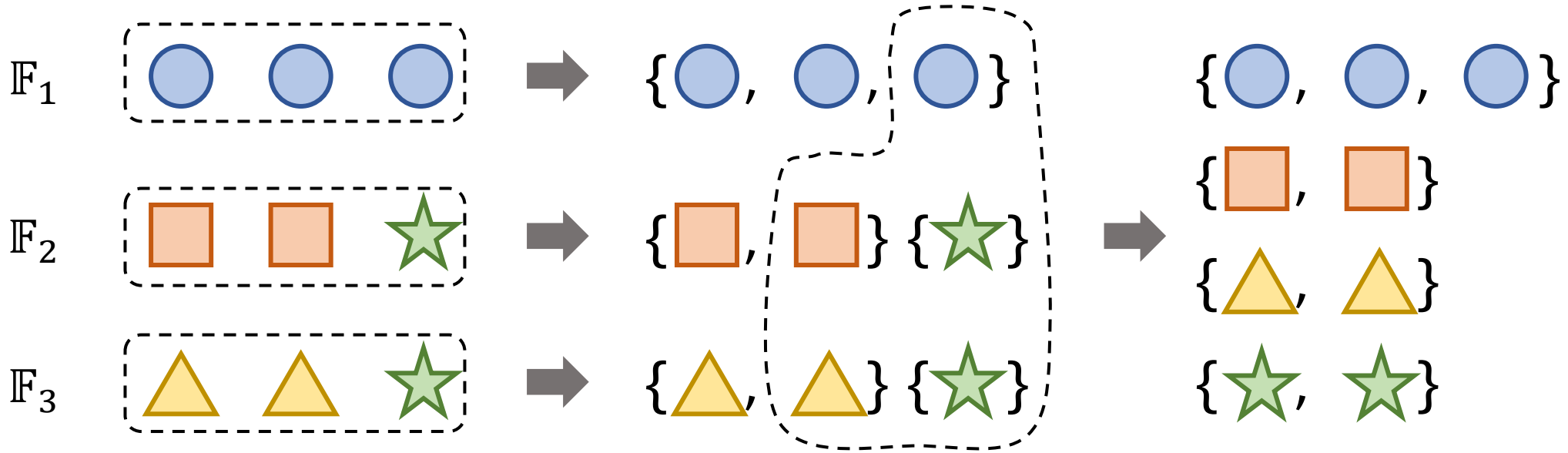
Fingerprint-Assisted Co-Location Verification



Fingerprint-Assisted Co-Location Verification



Fingerprint-Assisted Co-Location Verification



4 batch tests instead of $9 \times 8/2 = 36$ pairwise tests

More discussion in the paper

Host Fingerprints are Highly Accurate

- Gather fingerprints from 800 container instances
- Experiment is repeated 15 times in three data centers

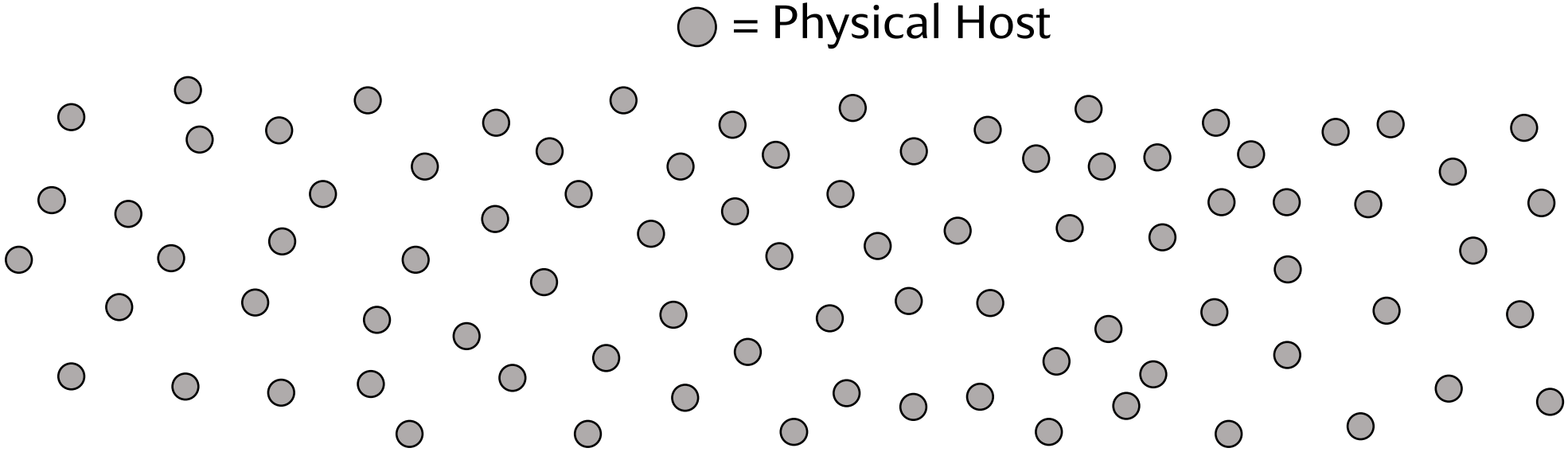
For each pair of container instances

- False positive (FP): same fingerprints but not co-located
- False negative (FN): different fingerprints but co-located

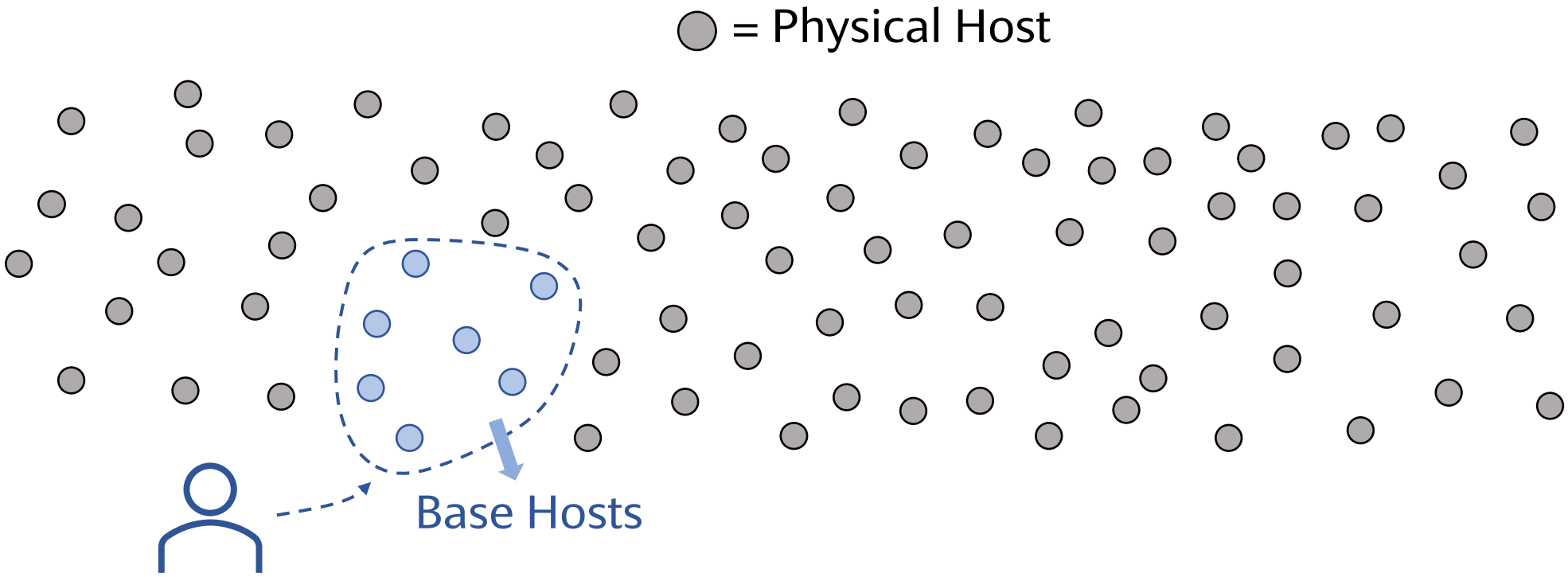
Average false negative rate: 0.00%
Average false positive rate: 0.02%

14 out of 15 measurements generate perfect fingerprints (no FP nor FN)

Understanding Instance Placement Policy



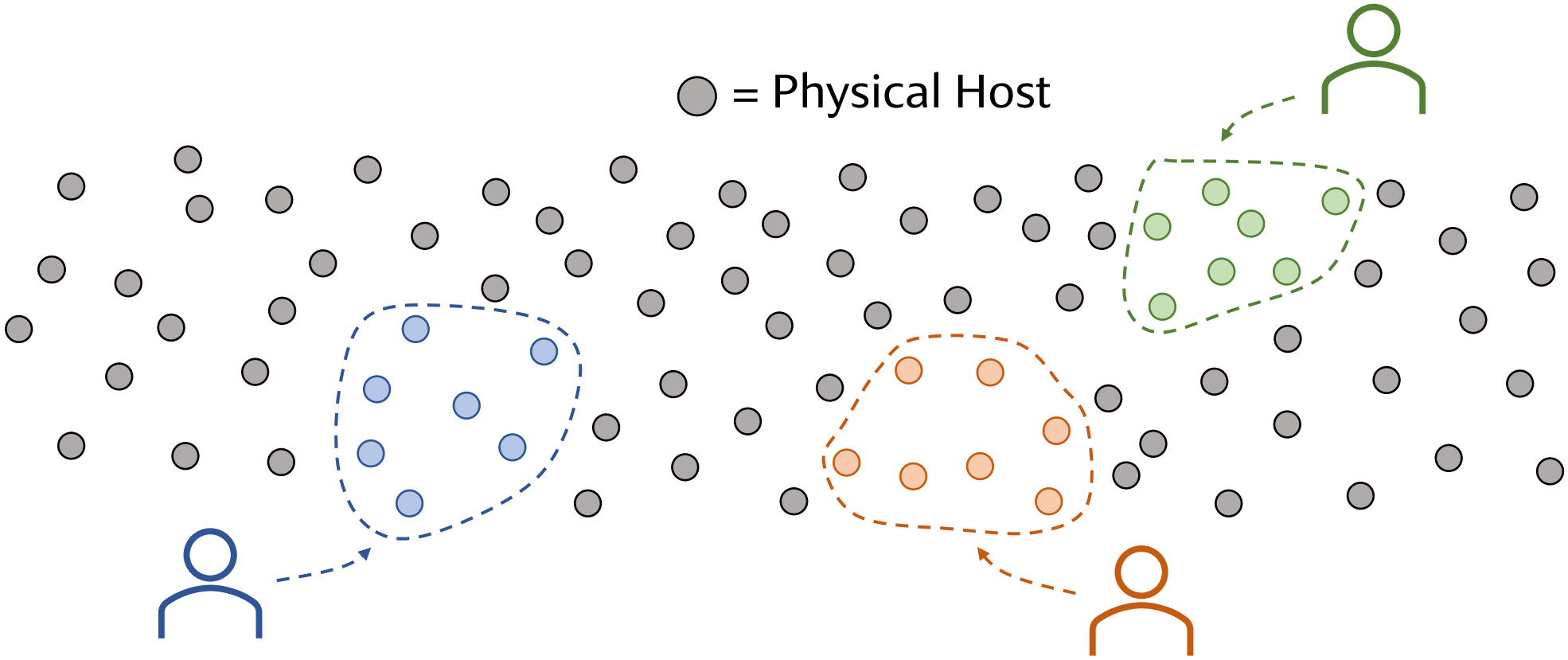
Base Hosts



Cloud Run prefers a specific set of hosts for instances owned by the same account

Possible cause: affinity scheduling to reduce communication overhead

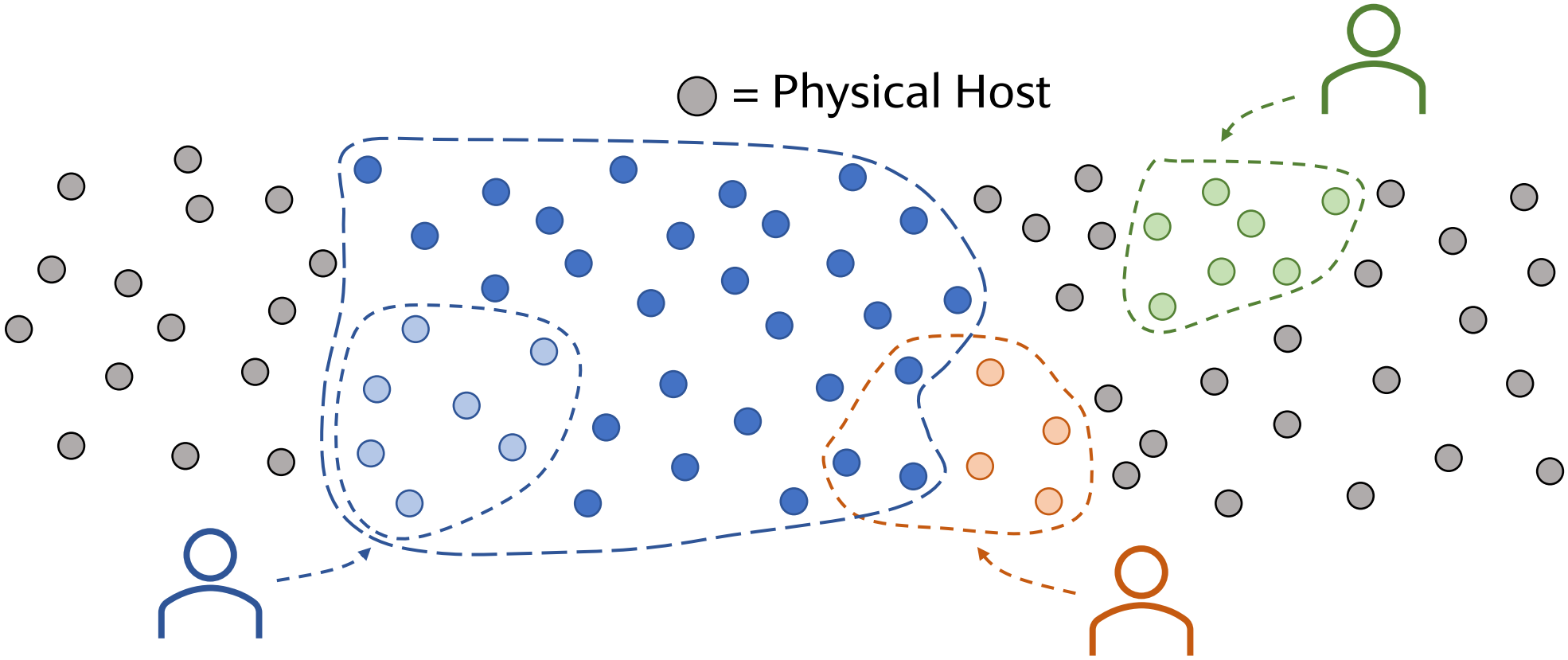
Different Accounts Have Different Base Hosts



Different accounts (usually) have different base hosts

Double-edged sword!

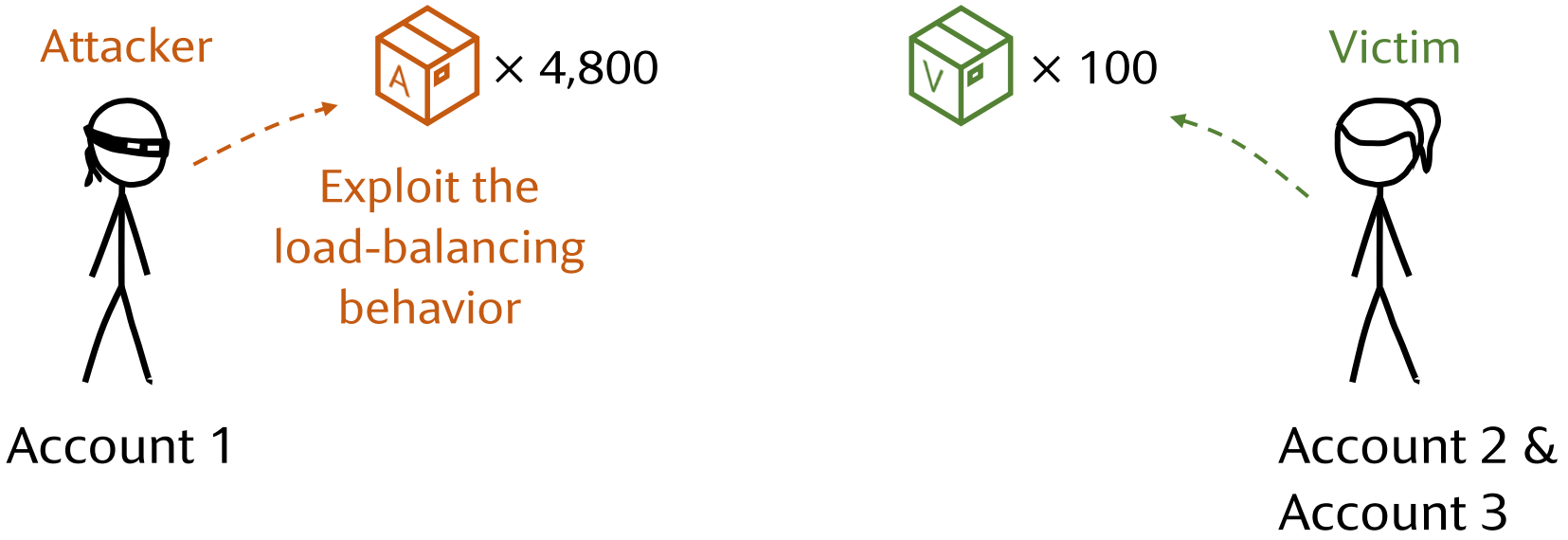
Load Balancing



High container usage can trick Cloud Run to spread instances across many hosts

☺ Allow attackers to go beyond their base hosts

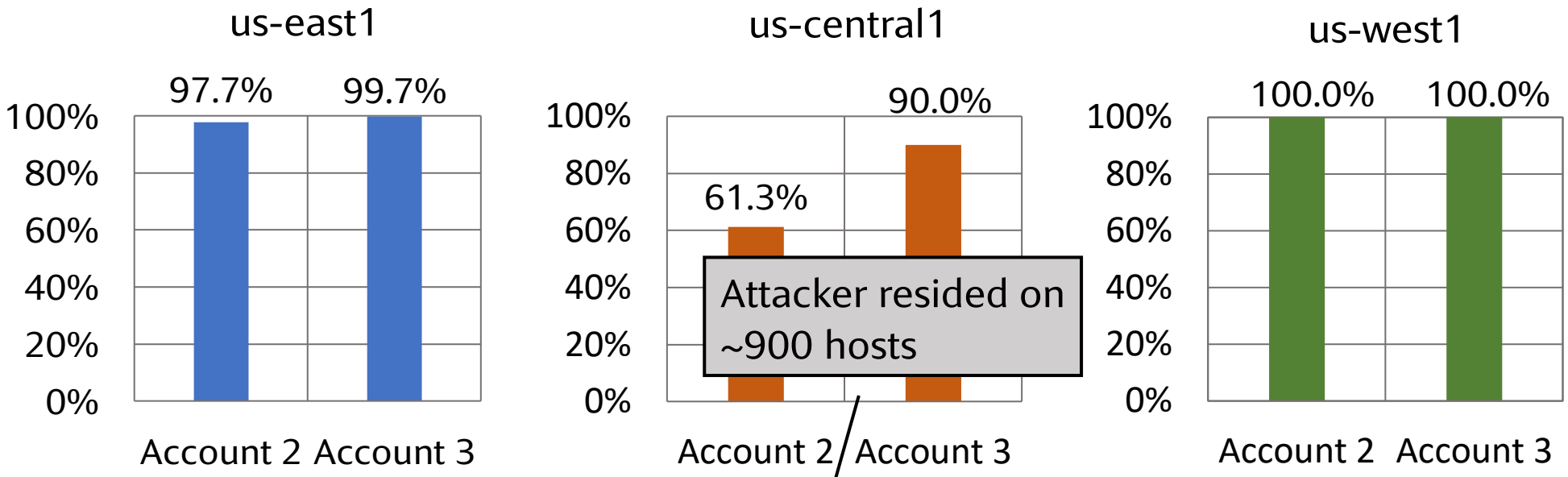
Co-Location with Victims



Victim Coverage: percentage of victim instances that are co-located with the attacker

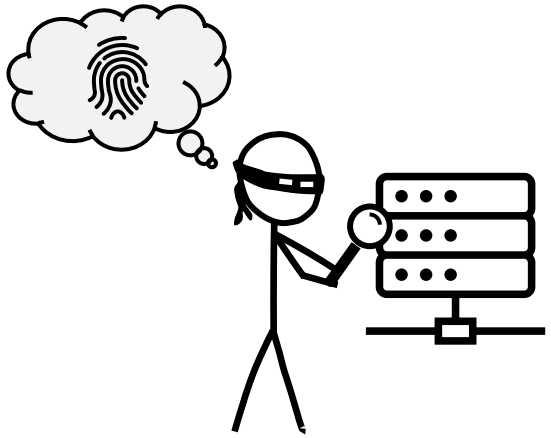
Co-Location with Victims

Average Victim Instance Coverage (3 repetitions in each region)

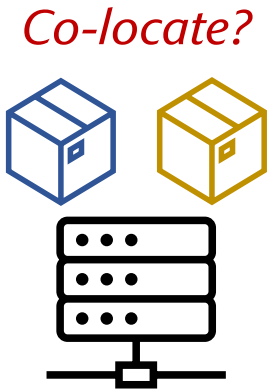


Attack Cost:	~24 USD	~23 USD	~27 USD

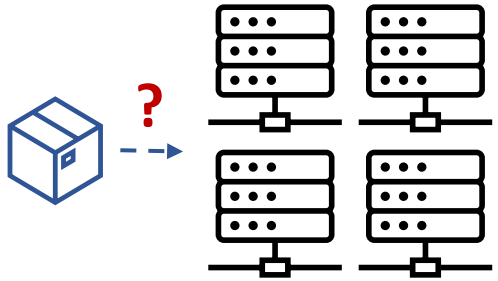
Conclusions



1. Accurate
Host Fingerprinting



2. Inexpensive
Co-Location Test



3. Exploitable
Placement Behavior

100% probability of co-locating with at least one victim instance

Co-locate with 61%-100% of victim instances in three major data centers