6.5810: Serverless + Isolation

Adam Belay <abelay@mit.edu>

MIT CSAIL

Serverless computing

- A new cloud programming model
- Key idea: Building applications without thinking about servers
- Function as a service (FaaS): Run a simple code function, let the cloud provider decide where and how to run it
- Typically, the function must be short (a few seconds or less) and consume relatively few resources (e.g., one core, 2GB RAM)
  - Makes it easier for cloud provider to pack instances
- Scale automatically; pay per use
- Consequence: Multiple tenants on each machine

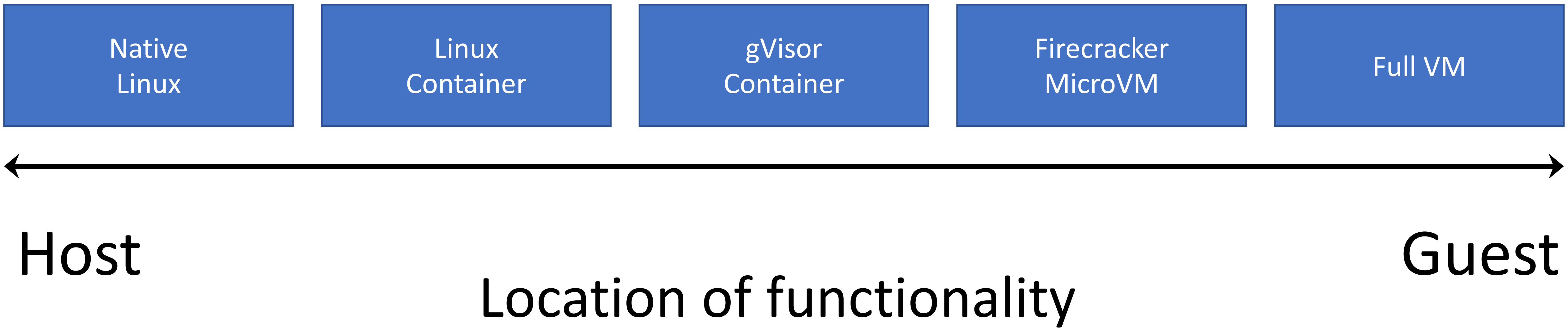
Agenda today

- Discuss the isolation and security aspect of serverless
- Explore new and recent ways of securing cloud applications
  - gVisor and Firecracker
- Review the solutions to lab 1
- Lab 3 will be assigned later today

Isolation schemes studied in the paper

- Native Linux: System call boundary determines isolation
- Linux containers: Same, but each container has a separate namespace maintained by the kernel (e.g., a different filesystem)
- gVisor Containers: OS functionality implemented as a library OS inside a Linux process. Library then makes a narrow set of system calls.
- Firecracker: Stripped down VMs, heavily paravirtualized
- Full VM: Guest kernel operates like a normal, complete kernel

Spectrum of OS functionality



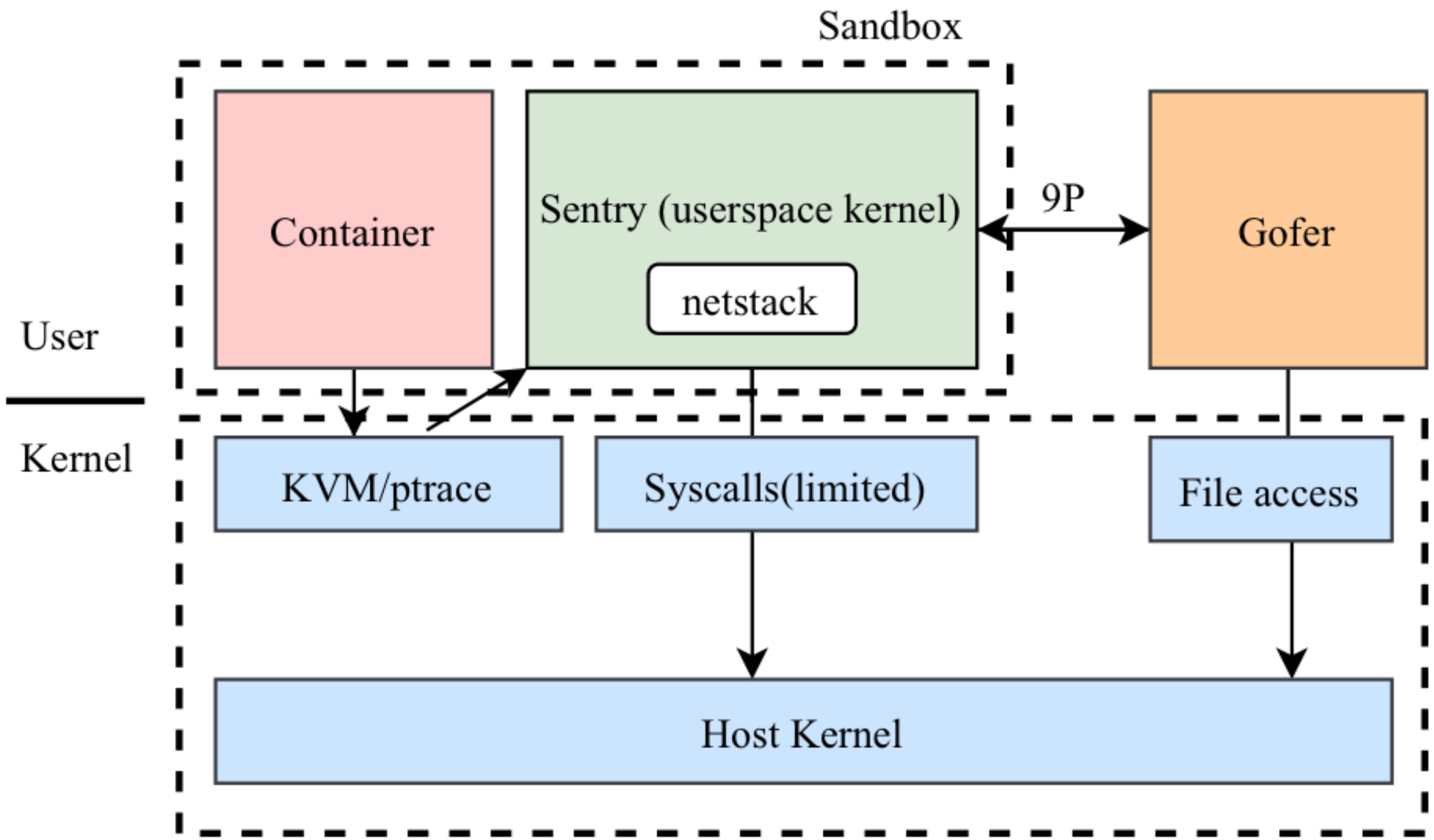
What is an attack surface?

- The sum of the different vectors where an attacker can try to break the isolation of a system
- One way of thinking: System calls are the attack surface
- This paper: Code coverage is the attack surface?

Linux containers

- A normal Linux process mostly; large attack surface (all system calls)
- *cgroups* provide resource limits, performance isolation, etc.
- *chroot* provides separate filesystem namespace
- Tools like docker make it easy to bundle and manage containers

gVisor architecture



Blending Containers and Virtual Machines: A Study of Firecracker and gVisor. Anjali et. Al. VEE'20

gVisor components

- **Sentry**: A userspace kernel, written in Go
  - All system calls made by the application are redirected to the Sentry
  - The sentry implements most system calls itself (supports 237 calls)
  - However, it makes 53 system calls to the host to support its operation
  - Seccomp filter restricts access to these calls
- App never directly makes host system calls (must go through sentry)
  - Ptrace-mode: ptrace forwards syscalls to sentry
  - KVM-mode: trap and handle system calls, forward to sentry (faster)
- **Gofer**: Provides sentry with access to file system resources
  - The sentry cannot directly read or write any files

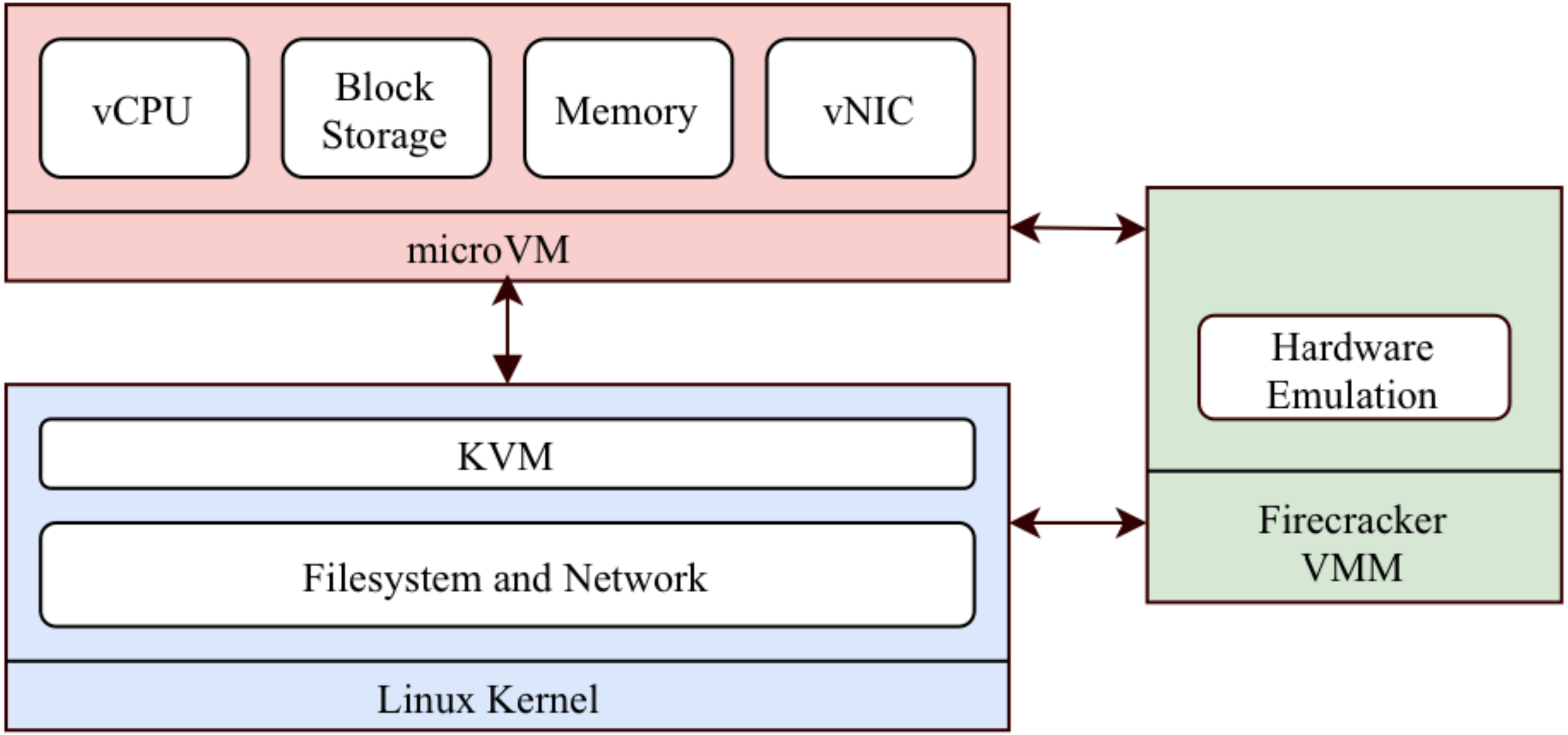
Seccomp filter

- Users can load custom code into the kernel without violating isolation
- Berkeley Packet Filter (BPF) provides a stripped-down, restricted assembly language that can be easily verified
  - Fixed-length instructions, 32-bit, 1 accumulator, 1 index register
- BPF code can be used to filter which system calls (and the arguments passed to them) are allowed

Example seccomp filter

```
struct sock_filter filter[] = {
  BPF_STMT(BPF_LD+BPF_W+BPF_ABS, syscall_nr),
  BPF_JUMP(BPF_JMP+BPF_JEQ+BPF_K, __NR_exit_group, 0, 1),
  BPF_STMT(BPF_RET+BPF_K, SECCOMP_RET_ALLOW),
  BPF_STMT(BPF_RET+BPF_K, SECCOMP_RET_KILL),
}
```

AWS Firecracker



Blending Containers and Virtual Machines: A Study of Firecracker and gVisor. Anjali et. Al. VEE'20

Firecracker components

- Uses a virtual machine, not a process (i.e., VT-x that we saw earlier)
- But still has somewhat of a Sentry, called the firecracker VMM
  - Manages storage and net I/O through virtio, a software I/O queue
- MicroVMs run an extremely stripped-down Linux distro
- More details on firecracker in upcoming lecture

Allowed system calls

Platform	Total allowed syscalls to the host kernel
LXC	all except 44
Firecracker	36
gVisor w/o host networking	53
gVisor w/ host networking	68

Table 1. Total number of system calls allowed out of 350

Blending Containers and Virtual Machines: A Study of Firecracker and gVisor. Anjali et. Al. VEE'20

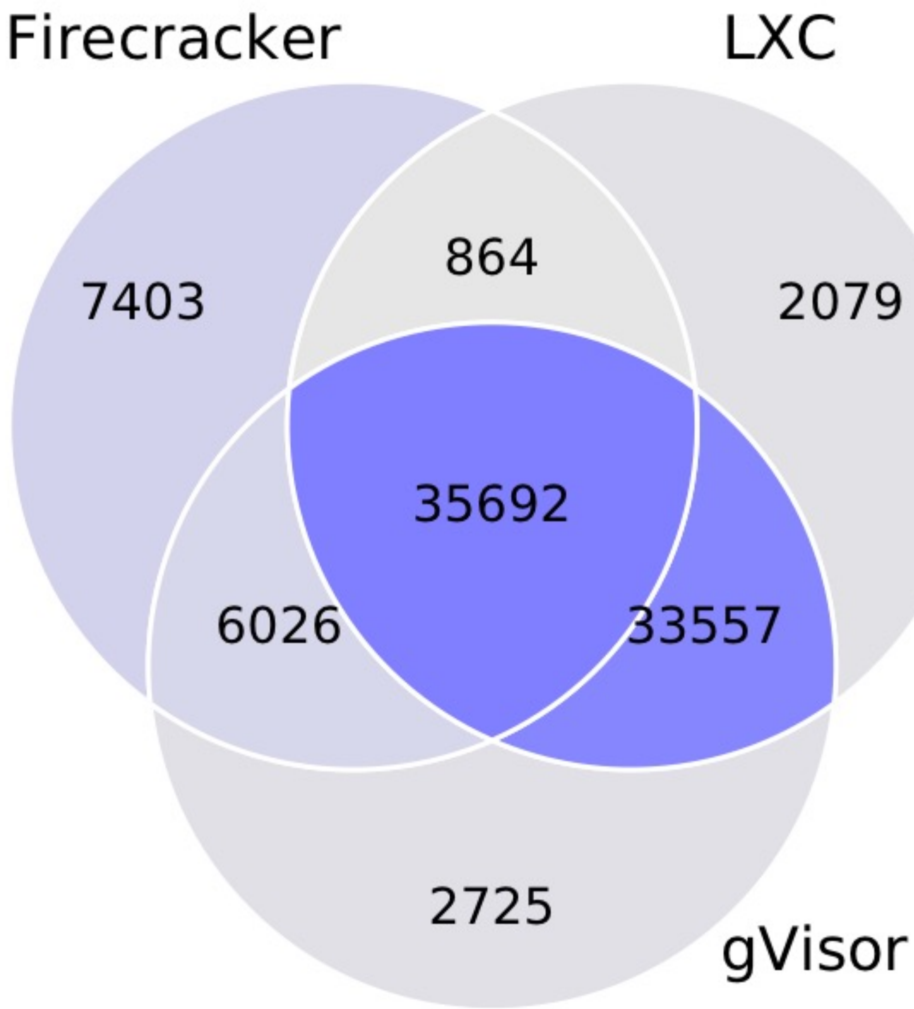
Code coverage

	Host	Firecracker	LXC	gVisor
Lines	63,163	77,392	90,595	91,161
Coverage	7.83%	9.59%	11.23%	11.31%

Table 2. Union of line coverage across all workloads out of 806,318 total lines in the Linux kernel.

Blending Containers and Virtual Machines: A Study of Firecracker and gVisor. Anjali et. Al. VEE’20

Code coverage venn diagram



Blending Containers and Virtual Machines: A Study of Firecracker and gVisor. Anjali et. Al. VEE’20

Networking bandwidth

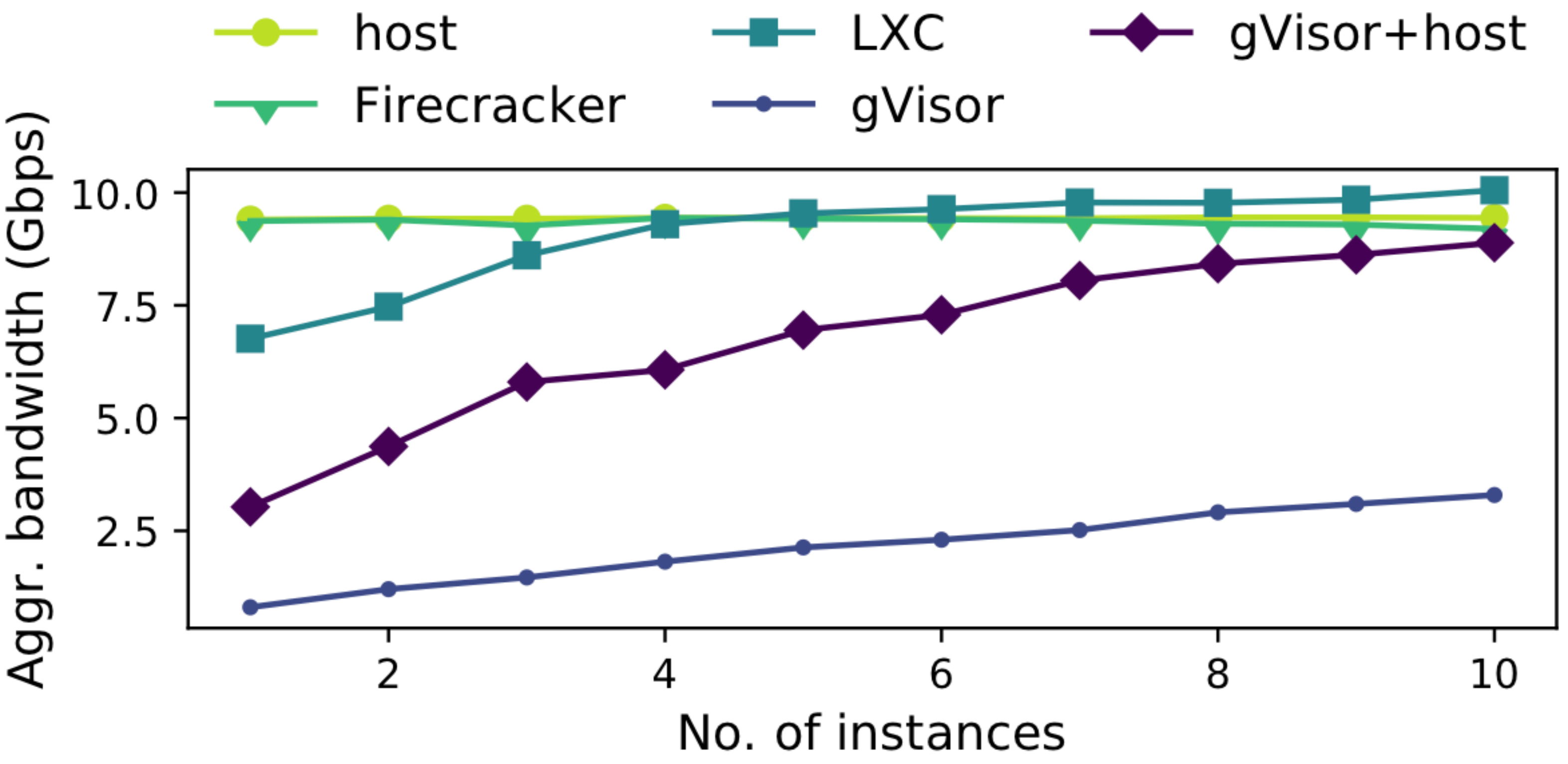


Figure 8. Aggregate Network Bandwidth

Blending Containers and Virtual Machines: A Study of Firecracker and gVisor. Anjali et. Al. VEE’20

Network latency

	Host	Firecracker	LXC	gVisor
RTT (μs)	146	371	149	319

Table 3. Round-trip time

Blending Containers and Virtual Machines: A Study of Firecracker and gVisor. Anjali et. Al. VEE'20

Memory management

- Two very different strategies
- gVisor’s sentry allocates memory in 16MB chunks using mmap()
- Firecracker’s guest manages its own guest-physical memory
  - But VMM must still trap and fill pages

Memory allocation overhead

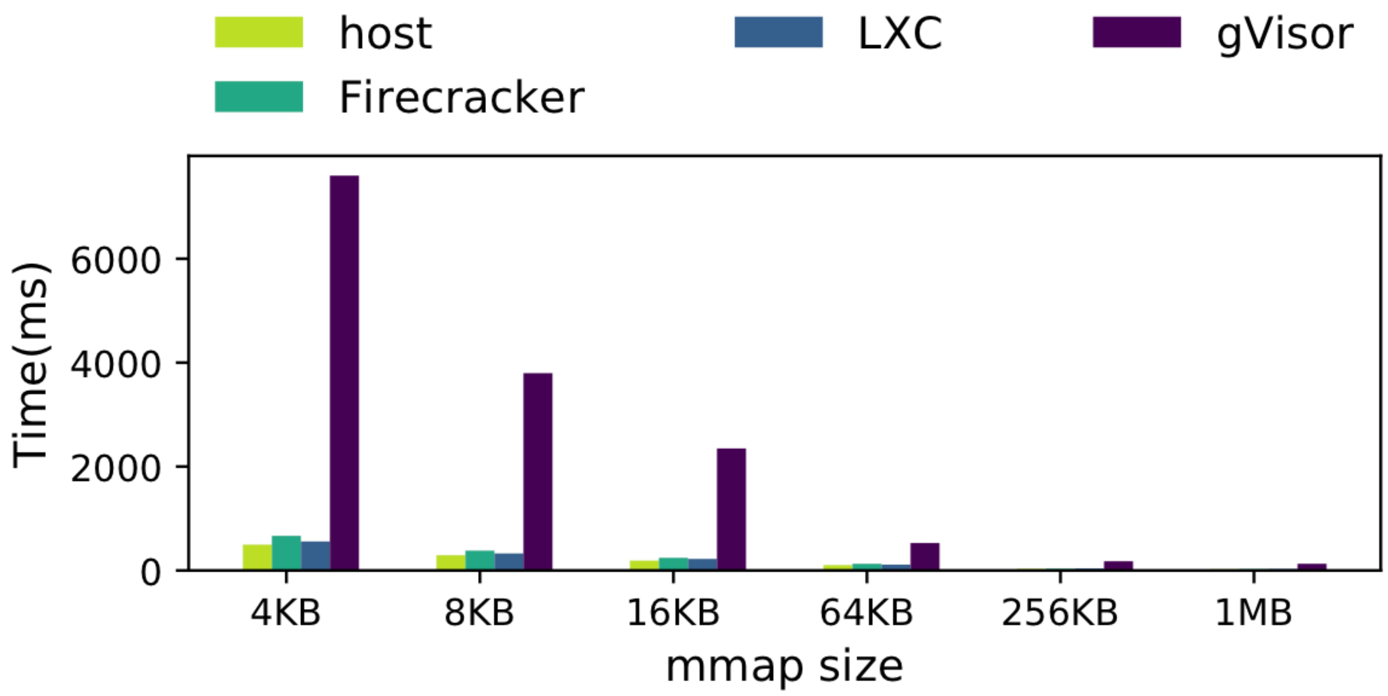


Figure 16. Total allocation+unmap time for 1GB

Blending Containers and Virtual Machines: A Study of Firecracker and gVisor. Anjali et. Al. VEE'20

What properties are desirable?

1. **Isolation:** The attack surface should be minimized
2. **Density:** Must be able to run as many instances as possible
3. **Performance:** Kernel overhead should be minimized; I/O performance should be fully exposed
4. **Compatibility:** Should be able to run unmodified applications

Debate: How are we doing so far?

- Isolation / Density / Performance / Compatibility
- gVisor, Firecracker, LXC, Host Linux?

Conclusion

- Existing isolation mechanisms, surprisingly, increase the amount of code that is typically executed
- But they decrease the amount of code that *could* be executed
- Firecracker guests access I/O at a lower level, mostly yielding less redundancy and better performance (relative to gVisor)
- Trapping system calls is costly for gVisor (even with KVM)
- No system performs well relative to kernel bypass

- We're building a better sandbox; come talk to us about final projects