Blending Containers and Virtual Machines: A Study of Firecracker and gVisor

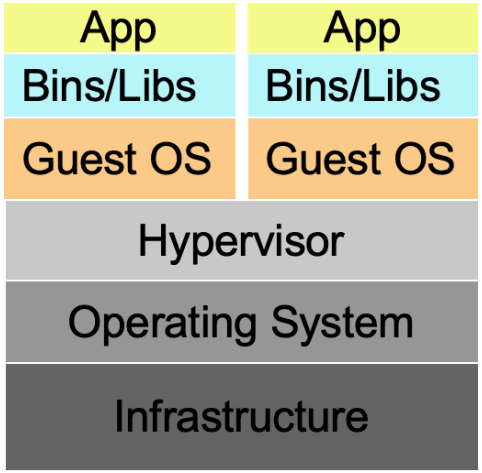
Anjali

Tyler Caraza-Harter

Michael M. Swift

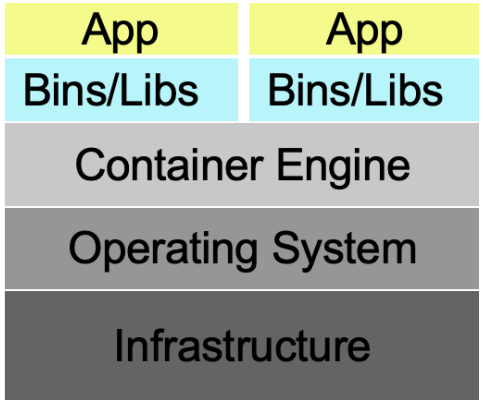
Department of Computer Sciences
University of Wisconsin-Madison

Existing (legacy) solutions



Hypervisor based virtualization

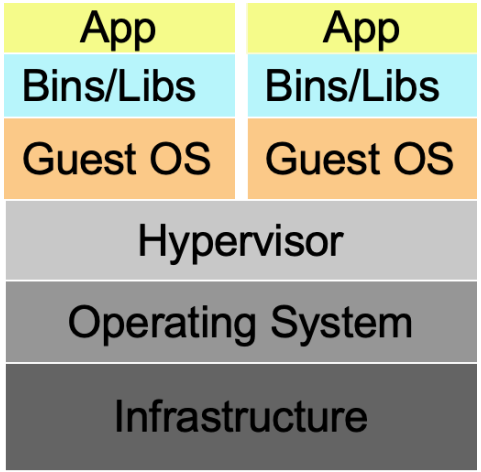
- Different kernel
- High boot time
- High memory footprint
- Strong isolation



Container virtualization

- Same kernel
- Fast boot time
- Low memory footprint
- Weak isolation

Existing (legacy) solutions

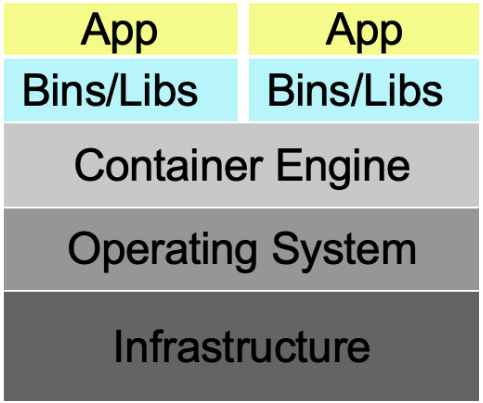


Firecracker

gVisor

Nabla Containers

kata containers



Hypervisor based virtualization

- Different kernel
- High boot time
- High memory footprint
- Strong isolation

Container virtualization

- Same kernel
- Fast boot time
- Low memory footprint
- Weak isolation

The dilemma

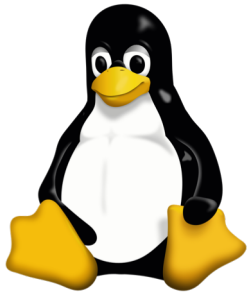
Application Developer

- Which has the best performance?
- How vulnerable are they to attacks from other containers?

Isolation Platform Developer

- How to design the architecture to minimize dependency on the host?

Kernel Developer



- How to streamline kernel support for isolation platforms?

Our work

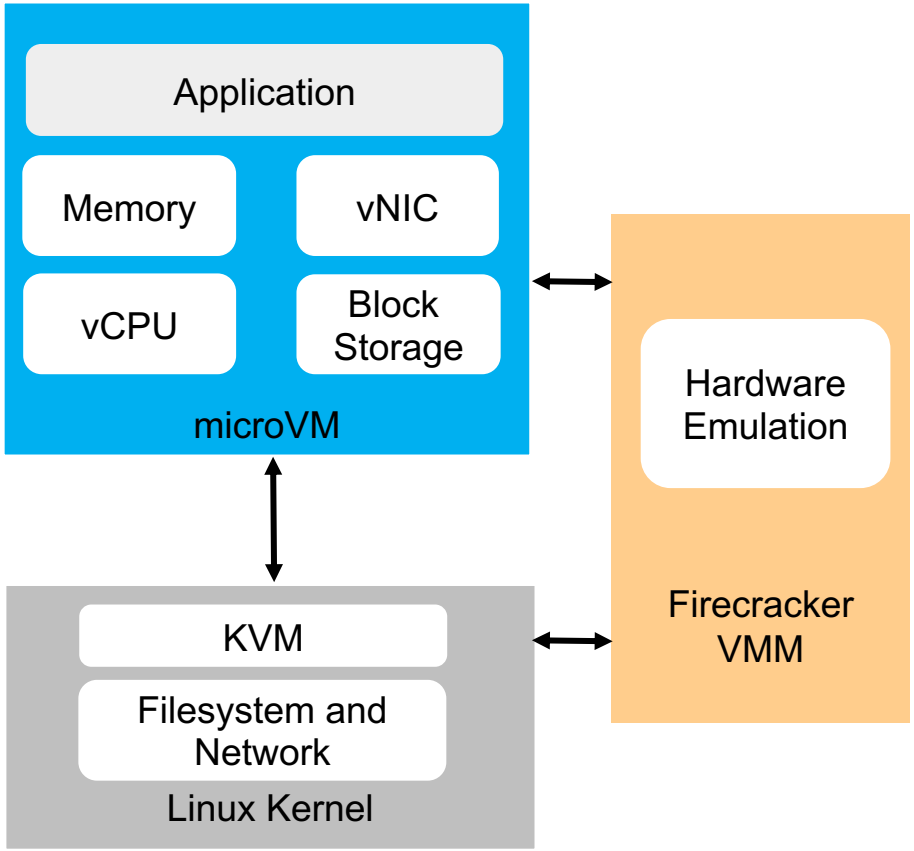
- Initial study
  - Compare properties of three secure isolation platforms
    - Linux containers
    - gVisor
    - Firecracker
  - Evaluate fundamental performance via microbenchmarks
  - Assess dependence on OS services via code tracing

Outline

- **Platform architecture**
  - Firecracker
  - gVisor
- Comparisons
  - Syscalls
  - CPU
  - Network
  - Overall

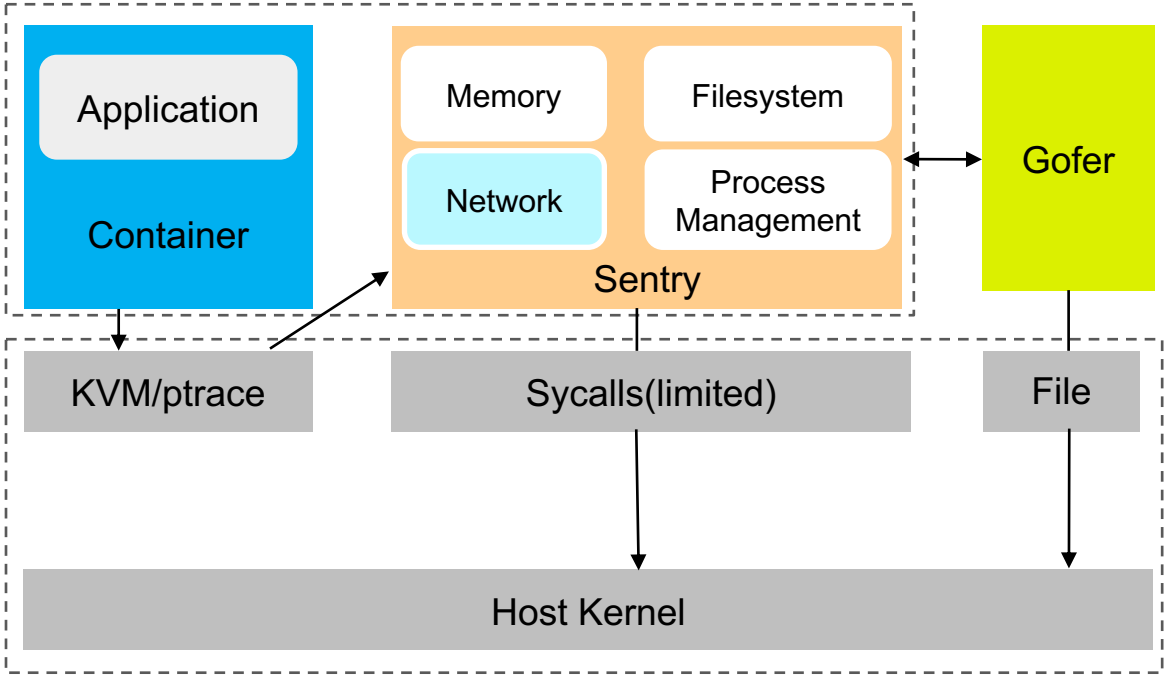
Platform Architecture: Firecracker

- Exports only 3 devices
- Limited system calls using SECure COMputing (seccomp) filters
- Written in Rust - type safe, memory safe, no unsafe C code etc.



Platform Architecture: gVisor

- Handles syscalls in sentry
- User space kernel written in Golang
- Sentry is heavily sandboxed
- Gofer for file access

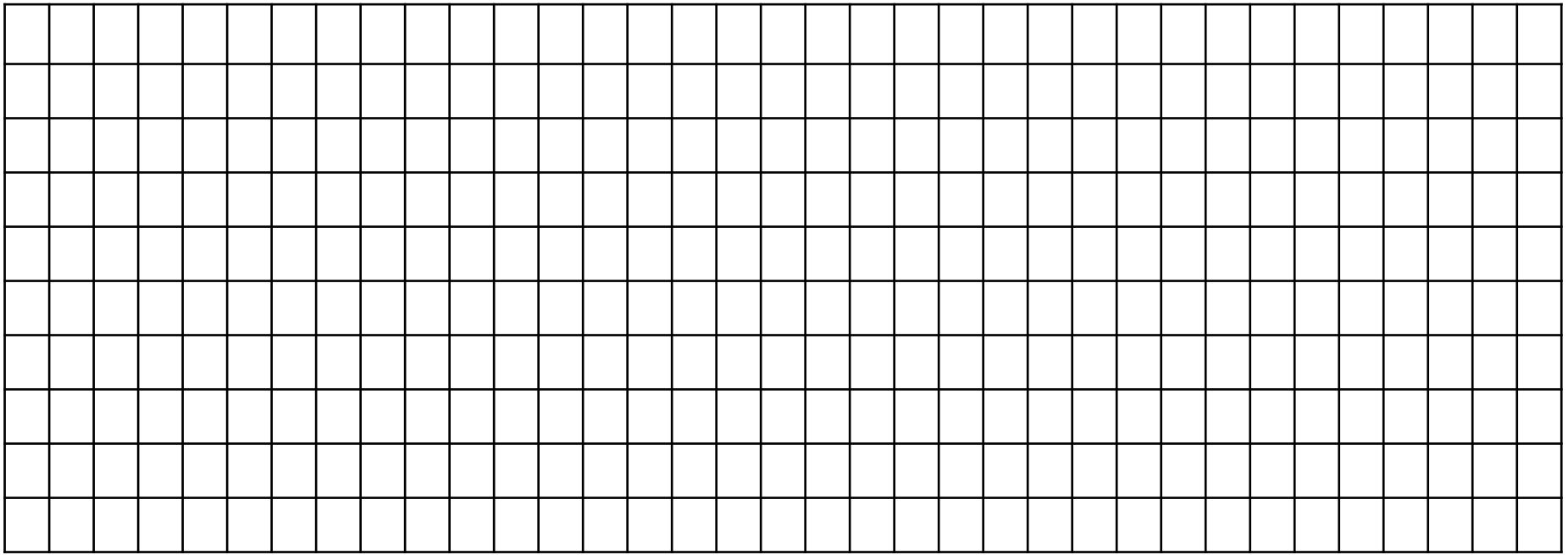


How are they different?

Firecracker	**gVisor**
Relies on guest and host kernel	Relies on Sentry and host kernel
Narrow syscall interface	Wider syscall interface
Low memory footprint	Low memory footprint
Type safe language	Type safe language

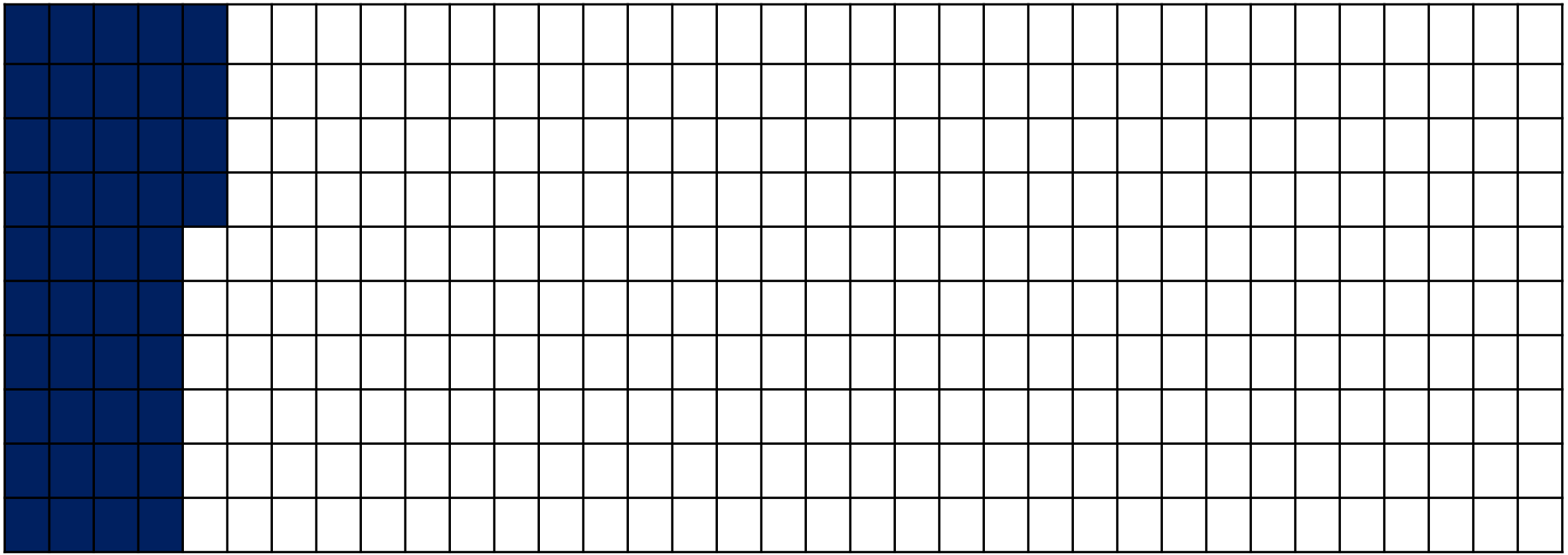
Both minimize interaction to the host kernel to enhance secure isolation by limiting syscalls interface.

Attack surface of syscall



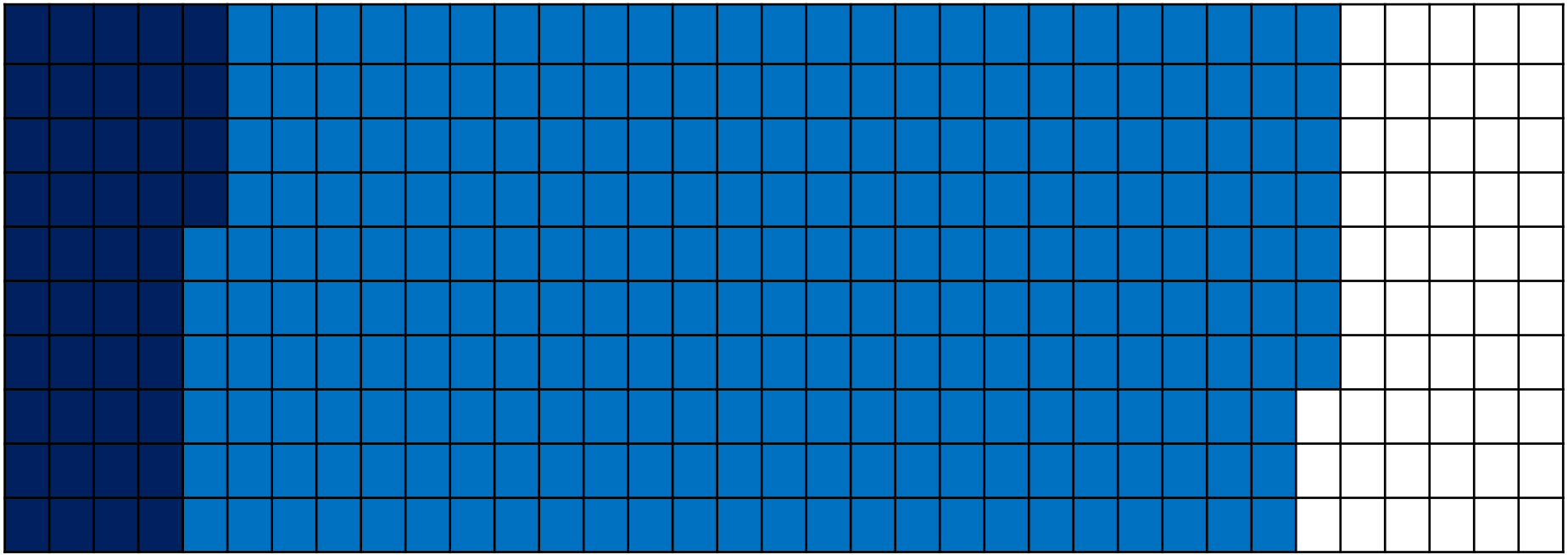
Attack surface of syscall

LXC

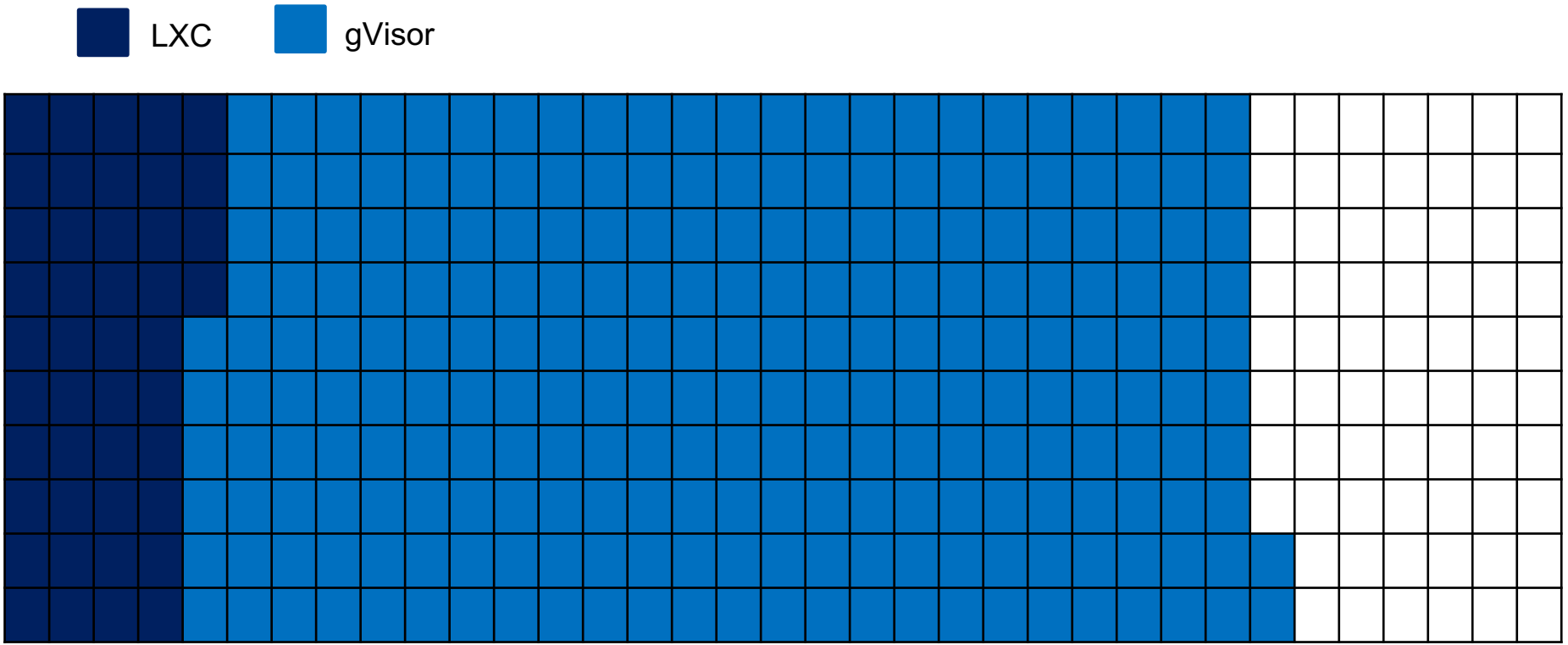


Attack surface of syscall

LXC gVisor

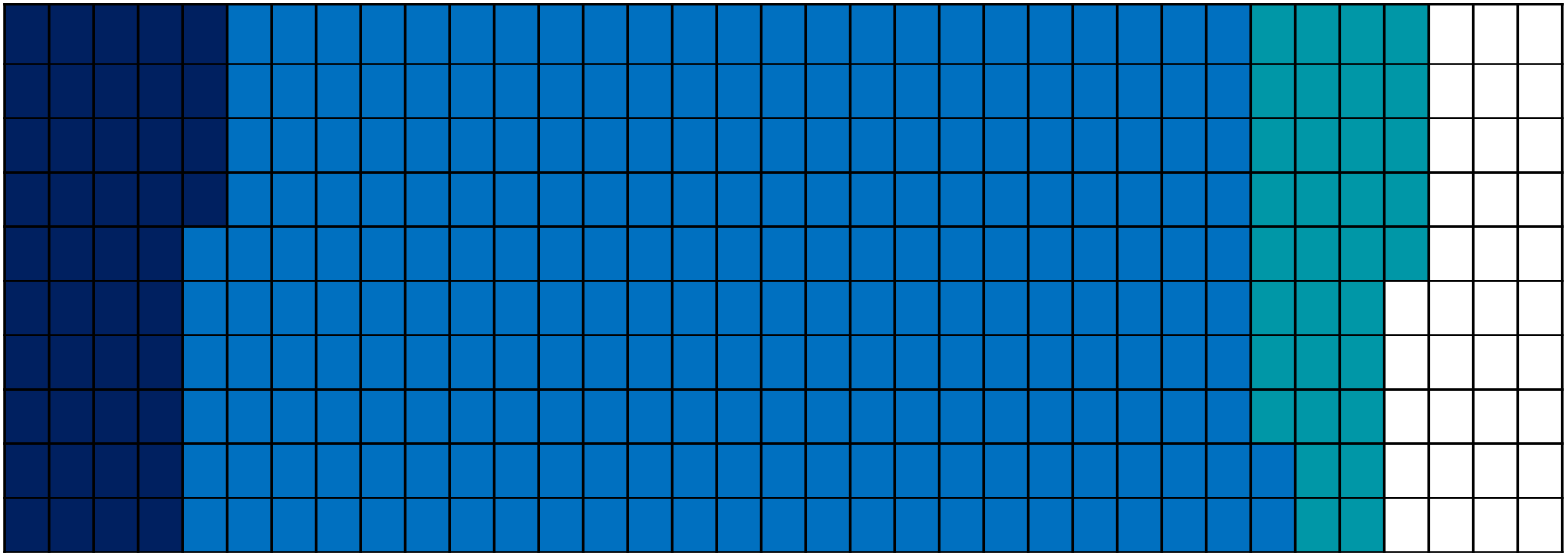


Attack surface of syscall



Attack surface of syscall

LXC gVisor Firecracker



Methodology

- Measure performance for microbenchmarks
  - CPU
  - Network
  - Memory
  - File I/O
- lcov to capture lines of source code executed for microbenchmarks
- Hardware - Cloudlab xl170 machine, a ten-core Intel E5-2640v4 running at 2.4 GHz, 64GBECC Memory (4x 16 GB DDR4-2400 DIMMs), Intel DC S3520 480 GB 6G SATA SSD and 10Gbps NIC.

lcov

LCOV - code coverage report

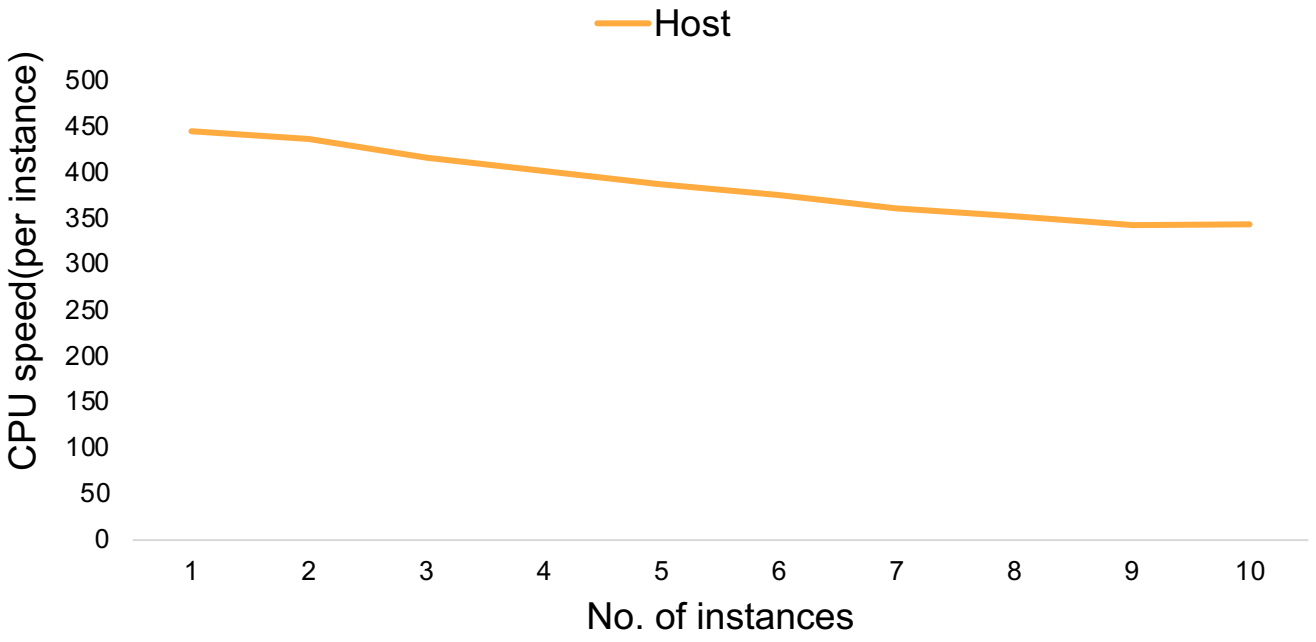
			Hit	Total	Coverage
Current view:	top level	**Lines:**	56945	806323	7.1 %
Test:	host_mem_10min_new.info	**Functions:**	6365	75512	8.4 %
Date:	2020-02-05 17:52:14	**Branches:**	28340	618467	4.6 %

Directory	Line Coverage		Functions		Branches	
arch/x86/crypto	2.9 %	19 / 660	1.2 %	1 / 86	3.1 %	9 / 288
arch/x86/entry	55.6 %	65 / 117	60.0 %	6 / 10	37.5 %	27 / 72
arch/x86/entry/vsyscall	0.0 %	0 / 106	0.0 %	0 / 11	0.0 %	0 / 98
arch/x86/events	3.9 %	37 / 954	3.0 %	3 / 99	1.3 %	10 / 783
arch/x86/events/amd	0.0 %	0 / 867	0.0 %	0 / 100	0.0 %	0 / 546
arch/x86/events/intel	1.8 %	96 / 5327	2.2 %	14 / 638	1.0 %	32 / 3332
arch/x86/hyperv	0.0 %	0 / 368	0.0 %	0 / 42	0.0 %	0 / 188
arch/x86/ia32	0.0 %	0 / 232	0.0 %	0 / 49	0.0 %	0 / 153
arch/x86/include/asm	35.1 %	579 / 1650	37.7 %	77 / 204	17.6 %	374 / 2120
arch/x86/include/asm/crypto	0.0 %	0 / 11	-	0 / 0	0.0 %	0 / 30
arch/x86/include/asm/e820	100.0 %	2 / 2	100.0 %	1 / 1	-	0 / 0
arch/x86/include/asm/fpu	59.4 %	57 / 96	71.4 %	10 / 14	40.9 %	18 / 44
arch/x86/include/asm/numachip	0.0 %	0 / 4	-	0 / 0	-	0 / 0
arch/x86/include/asm/trace	20.5 %	9 / 44	11.8 %	12 / 102	0.0 %	0 / 442
arch/x86/include/asm/vdso	100.0 %	2 / 2	-	0 / 0	-	0 / 0
arch/x86/include/asm/xen	0.0 %	0 / 120	0.0 %	0 / 9	0.0 %	0 / 98
arch/x86/kernel	5.7 %	642 / 11307	5.9 %	69 / 1170	3.0 %	212 / 7045

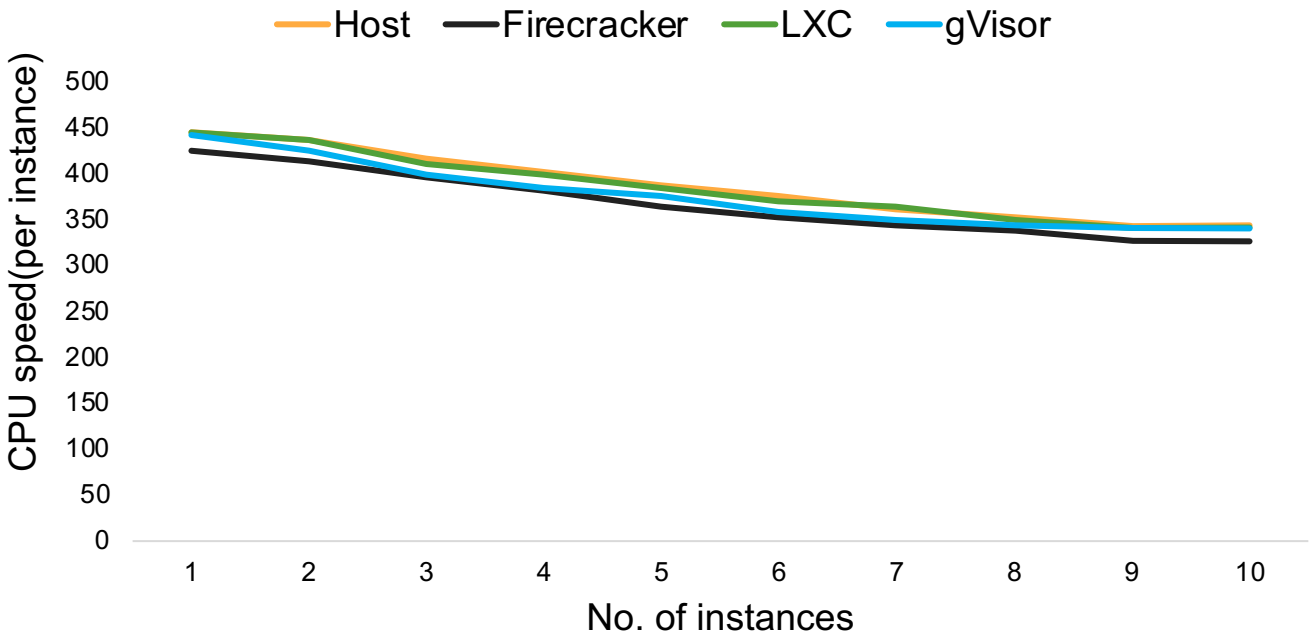
```
                    :          :              */
[ + + + - ]         :      310 :             if (exporter && exporter->anon_vma && !importer->anon_vma) {
      # # ]
                    :          :                     int error;
                    :          :
                    :        0 :                     importer->anon_vma = exporter->anon_vma;
                    :        0 :                     error = anon_vma_clone(importer, exporter);
        [ # # ]:             0 :                     if (error)
                    :          :                             return error;
                    :          :             }
                    :          :     }
                    :     4955 : again:
                    :     4955 :     vma_adjust_trans_huge(orig_vma, start, end, adjust_next);
                    :          :
        [ + + ]:          4955 :     if (file) {
                    :     4521 :             mapping = file->f_mapping;
                    :     4521 :             root = &mapping->i_mmap;
                    :     4521 :             uprobe_munmap(vma, vma->vm_start, vma->vm_end);
                    :          :
        [ - + ]:          4521 :             if (adjust_next)
                    :        0 :                     uprobe_munmap(next, next->vm_start, next->vm_end);
                    :          :
                    :          :             i_mmap_lock_write(mapping);
        [ + - ]:          4521 :             if (insert) {
                    :          :                     /*
                    :          :                      * Put into interval tree now, so instantiated pages
                    :          :                      * are visible to arm/parisc __flush_dcache_page
                    :          :                      * throughout; but we cannot insert into address
                    :          :                      * space until vma start or end is updated.
                    :          :                      */
                    :     4521 :                     __vma_link_file(insert);
                    :          :             }
                    :          :     }
```

General Findings

CPU: Sysbench Performance

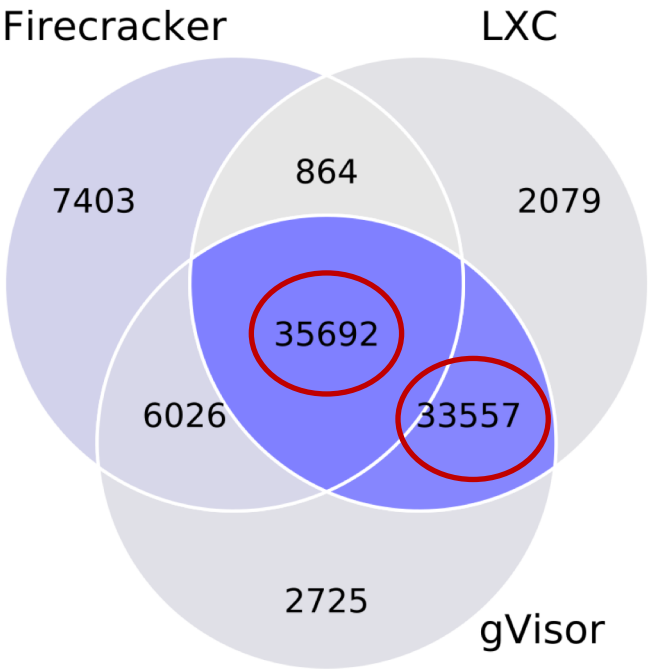


CPU: Sysbench Performance

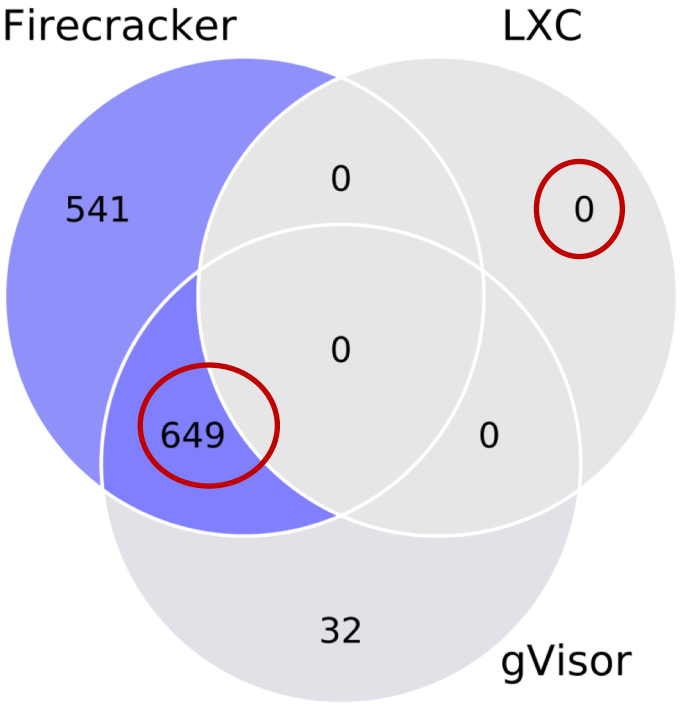


CPU: Coverage

Overall



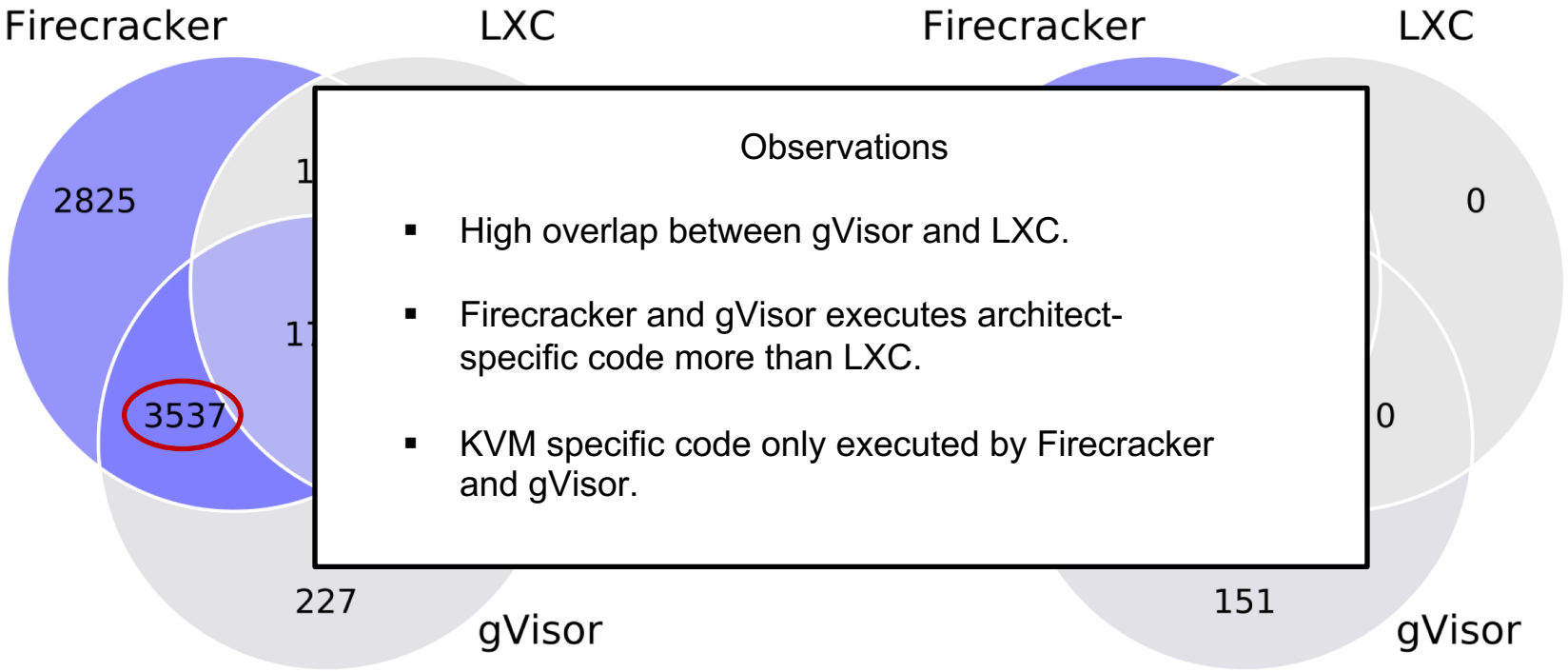
/virt



CPU: Coverage

/arch

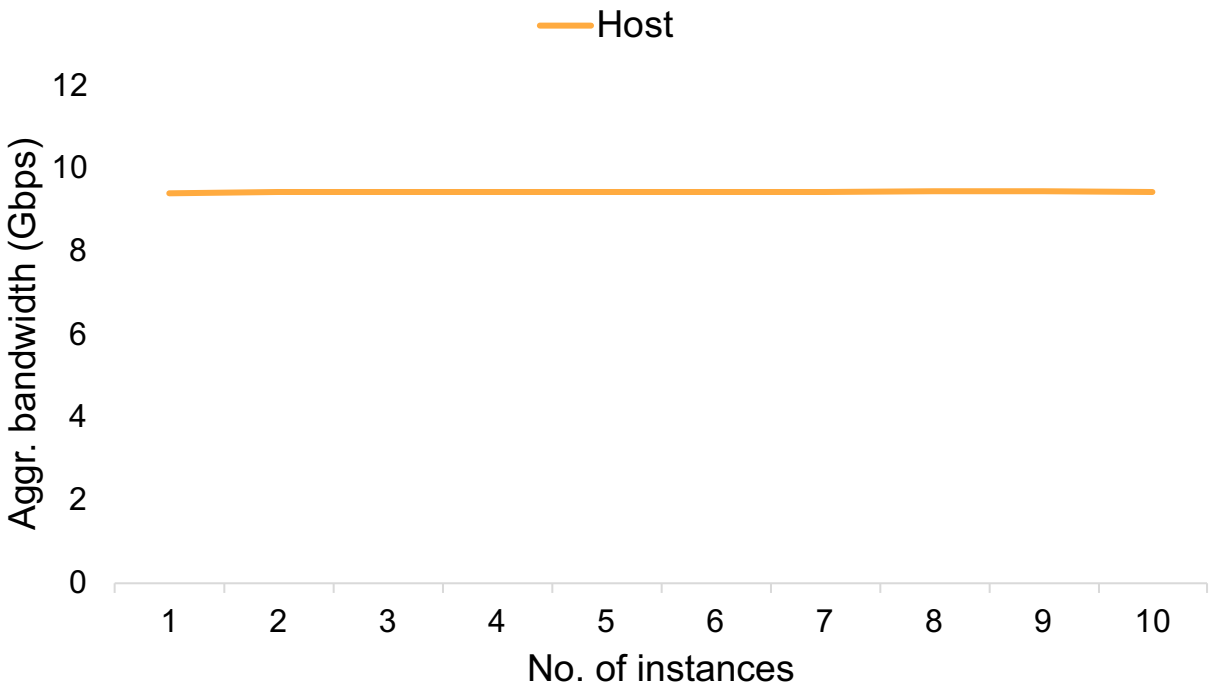
/arch/x86/kvm



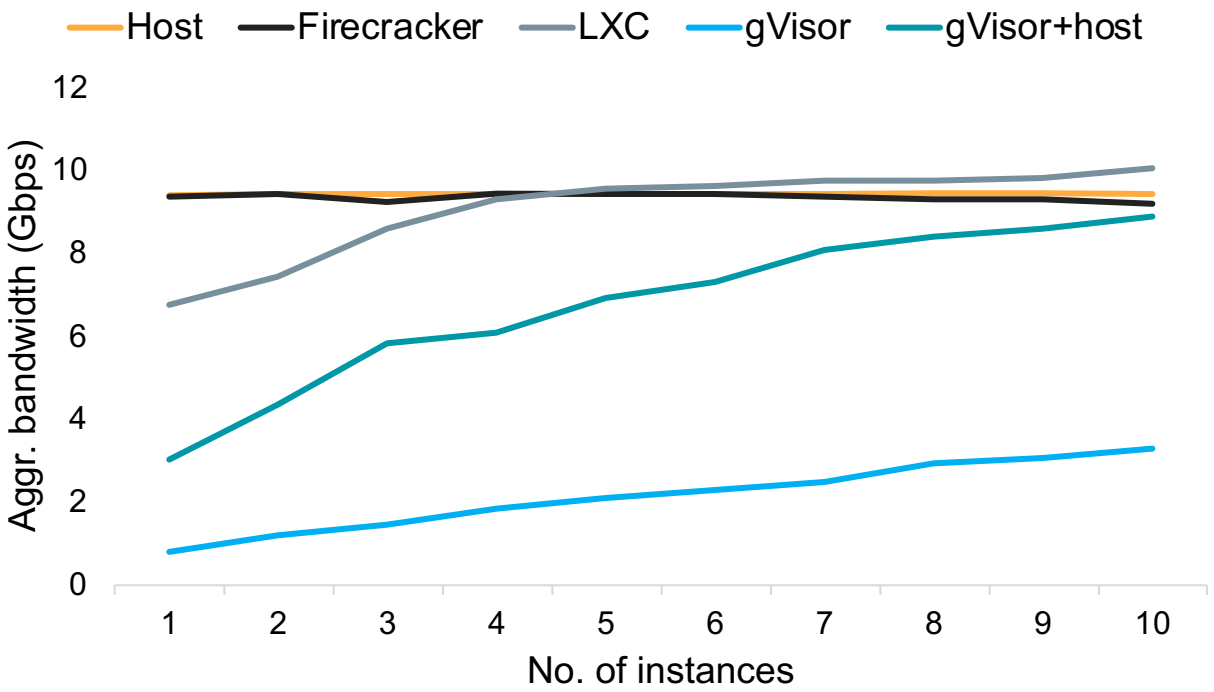
Observations

- High overlap between gVisor and LXC.
- Firecracker and gVisor executes architect-specific code more than LXC.
- KVM specific code only executed by Firecracker and gVisor.

Network: Performance

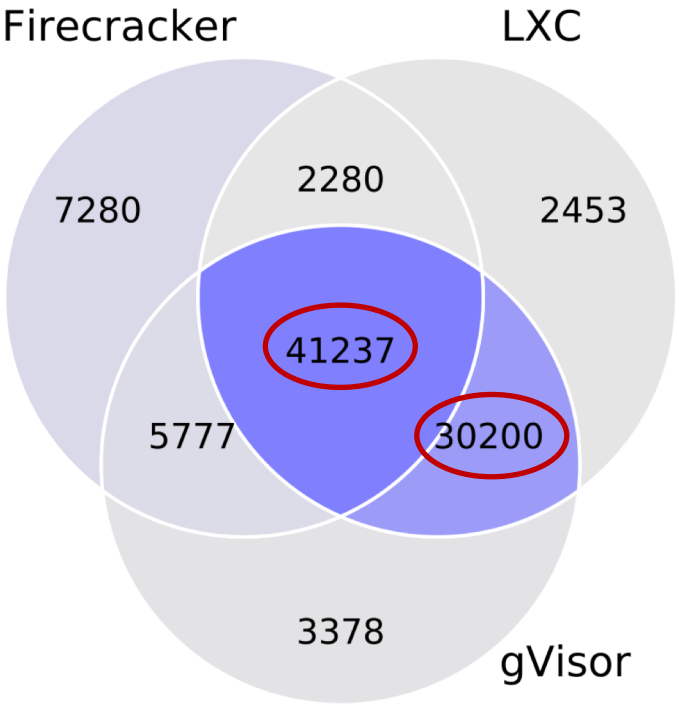


Network: Performance

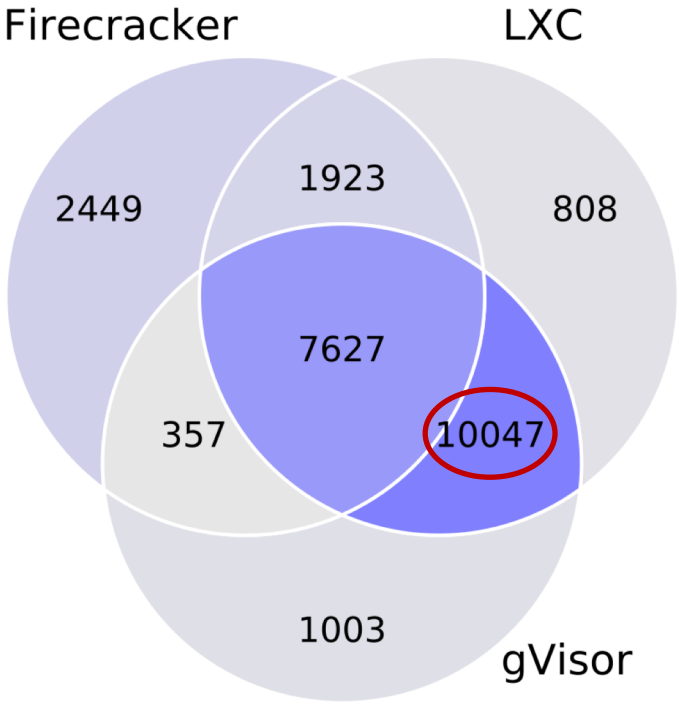


Network: Coverage

Overall

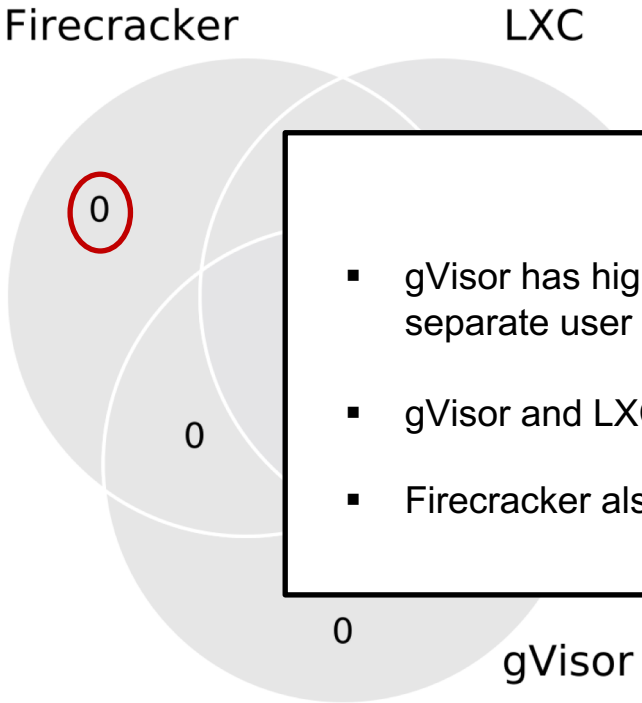


/net

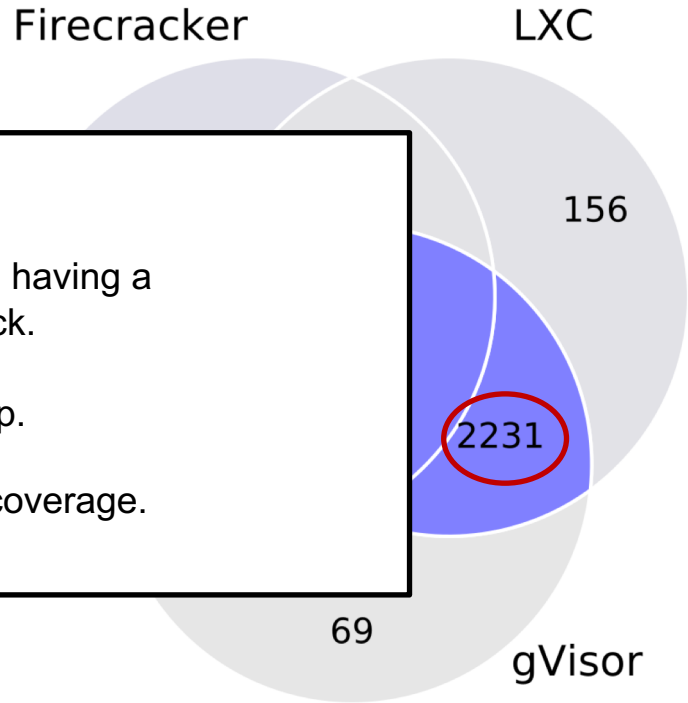


Network: Coverage

/net/bridge



/net/core



Observations

- gVisor has high coverage despite having a separate user space network stack.
- gVisor and LXC have high overlap.
- Firecracker also has substantial coverage.

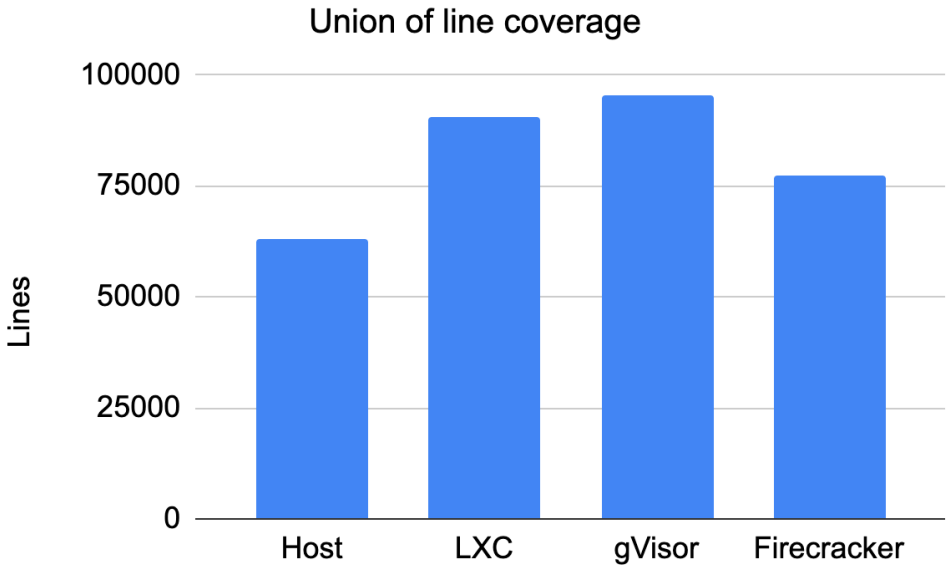
Invocation Frequency

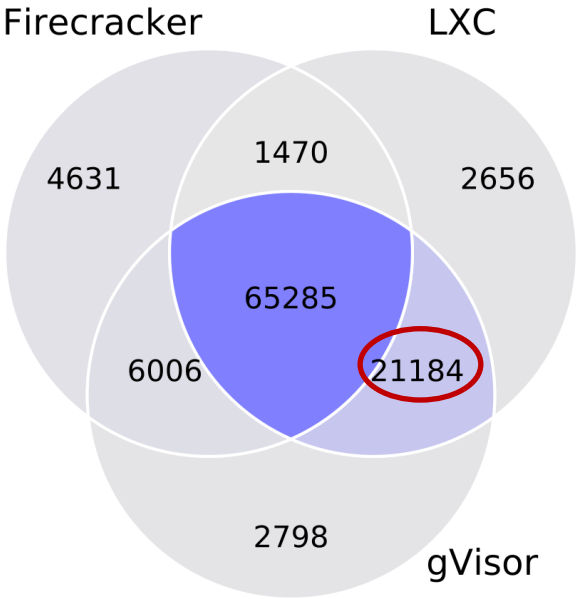
```
bool is_skb_forwardable(const struct net_device *dev,          1823
        const struct sk_buff *skb){
   unsigned int len;                                           1824
   if (!(dev->flags & IFF_UP))                                 1825
      return false;                                            1826
   len = dev->mtu + dev->hard_header_len +                     1827
        VLAN_HLEN;
   if (skb->len <= len)                                        1828
      return true;                                             1829
   if (skb_is_gso(skb))                                        1830
      return true;                                             1831
   return false;                                               1832
}                                                              1833
```

	Hits		
Lines	LXC	gVisor	Firecracker
1825	0.2 billion	5 million	0
1827	0.2 billion	5 million	0
1828	0.2 billion	5 million	0
1830	8 million	0.9 million	0

net/core/dev.c

Overall Coverage





Conclusion

- Neither gVisor nor Firecracker are best for all workloads
- Firecracker – High host kernel code footprint
- gVisor – High dependence on kernel functionality
- Optimize code paths

Next Steps

- Expand to other isolation platforms

Questions?

anjali@wisc.edu