improving debuggability of nommu code with UML

(why do we need nommu kernel in Linux ?)

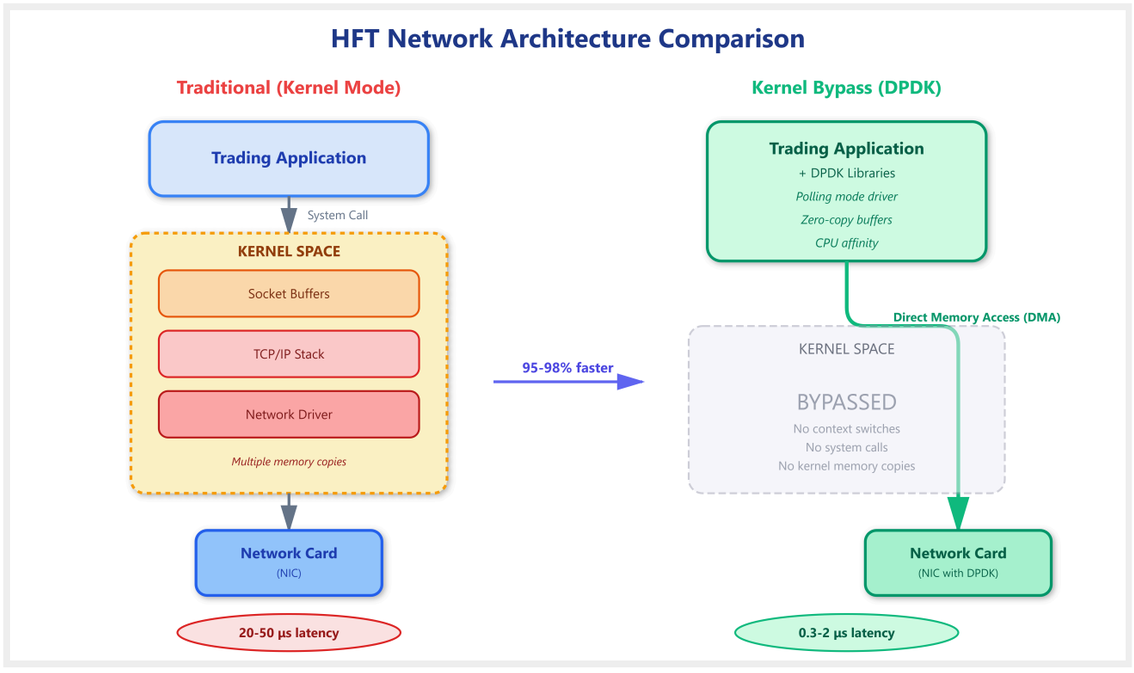
Hajime Tazaki (@thehajime)

Linux netdev conference 0x1a

Userspace network stacks

	year	lang	how	API	features	origina (if any)
lwip	(2001)	C	src-embedded	custom	v4,v6,ipfwd,tcp	scratch
Seastar	(2014)	C++17	static lib	custom	v4,tcp,dpdk	scratch
OSv	(2013)	C++/C	static lib	POSIX	v4,tcp	(freebsd
netstack	(2018)	golang	go pkg	custom	v4,v6,tcp	scratch
mTCP	(2014)	C	static lib	custom	v4,tcp,dpdk	scratch
rump	(2007)	C,asm	static/sh lib	POSIX	v4,v6,ipfwd,tcp	NetBSD
LKL	(2007?)	C,asm	static/sh lib	POSIX	v4,v6,ipfwd,tcp,dpdk	Linux
iip	(2023)	C	src-embeded	custom	v4,tcp,dpdk	scratch
f-stack	(2017?)	C,asm	static/sh lib	POSIX	v4,v6,ipfwd,tcp,dpdk	freebsd

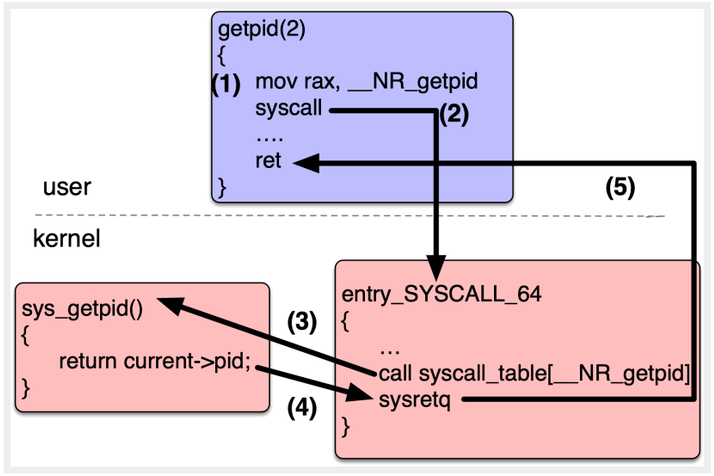
Userspace network stack (cont'd)



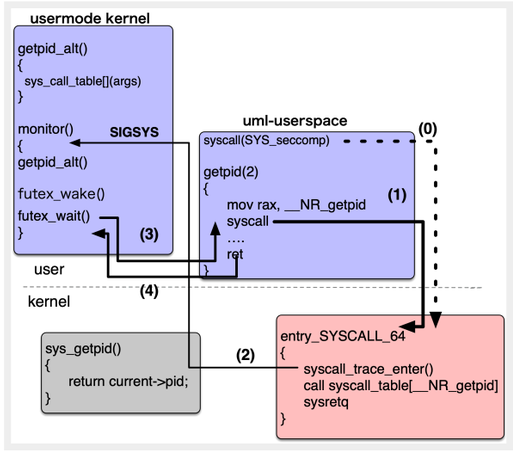
- implement network stack in userspace which usually implemented in kernel space
  - Why ? fast, lightweight, easy deployment (self-contained) !
  - Why Fast ? less syscall, less context switch
  - How ? simplify memory management, efficient device I/O, syscall hook

*ref: https://systemdr.systemdrd.com/p/high-frequency-trading-architecture

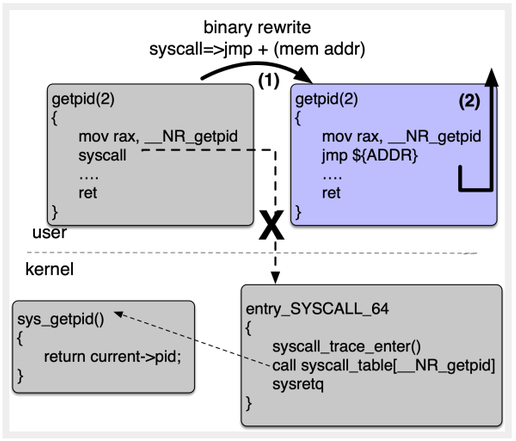
Why Fast ?



baremetal



uml(seccomp)



uml(nommu-seccomp)

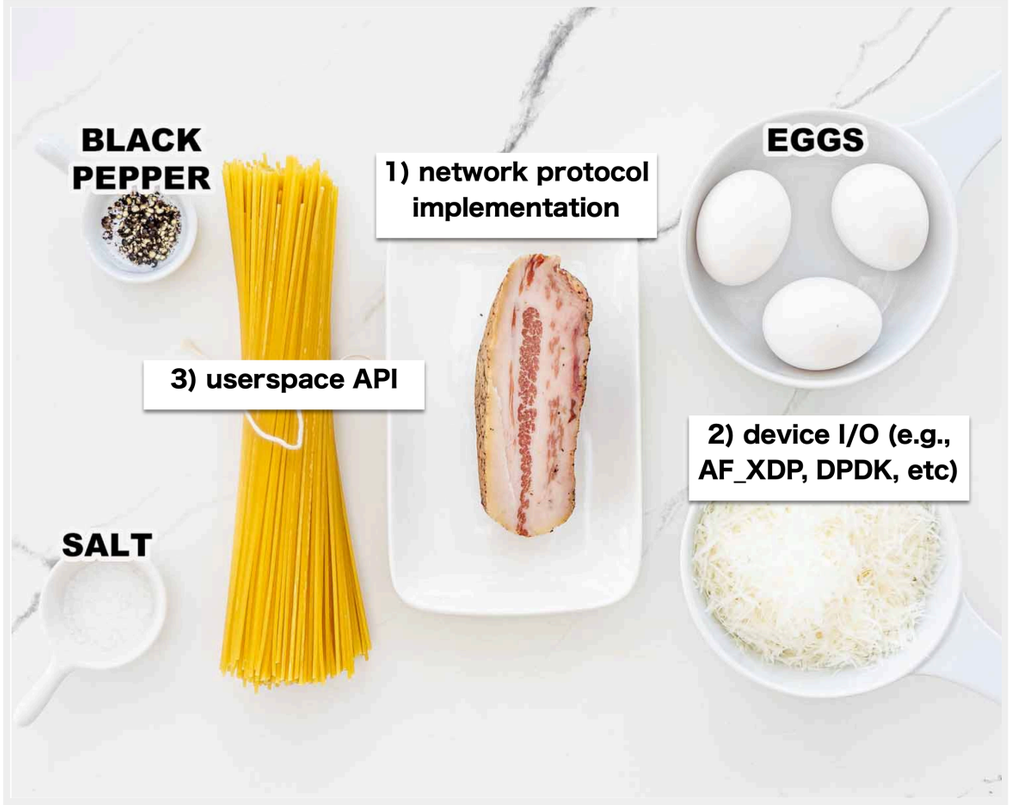
- single memory address space
  - simplifies memory operations
    - no kernel/user space separation, no page table
  - sacrificing various things (no longer virtual memory)
- e.g.) # of context switch
  - (conventional) UML == u/k *1, u/u *2 (via idle)
  - nommu UML: == 0

	native	**uml**	**uml-nommu**
getpid (nsec)	106	23655	172

Alternate options

- use full-scratch based userspace network stacks
  - maybe fast, API (ABI) compatible
  - no feature-rich implementations
- reduce # of syscalls
  - dedicated API (e.g., sendm{m,mm}sg, io_uring)
  - need to rewrite existing applications

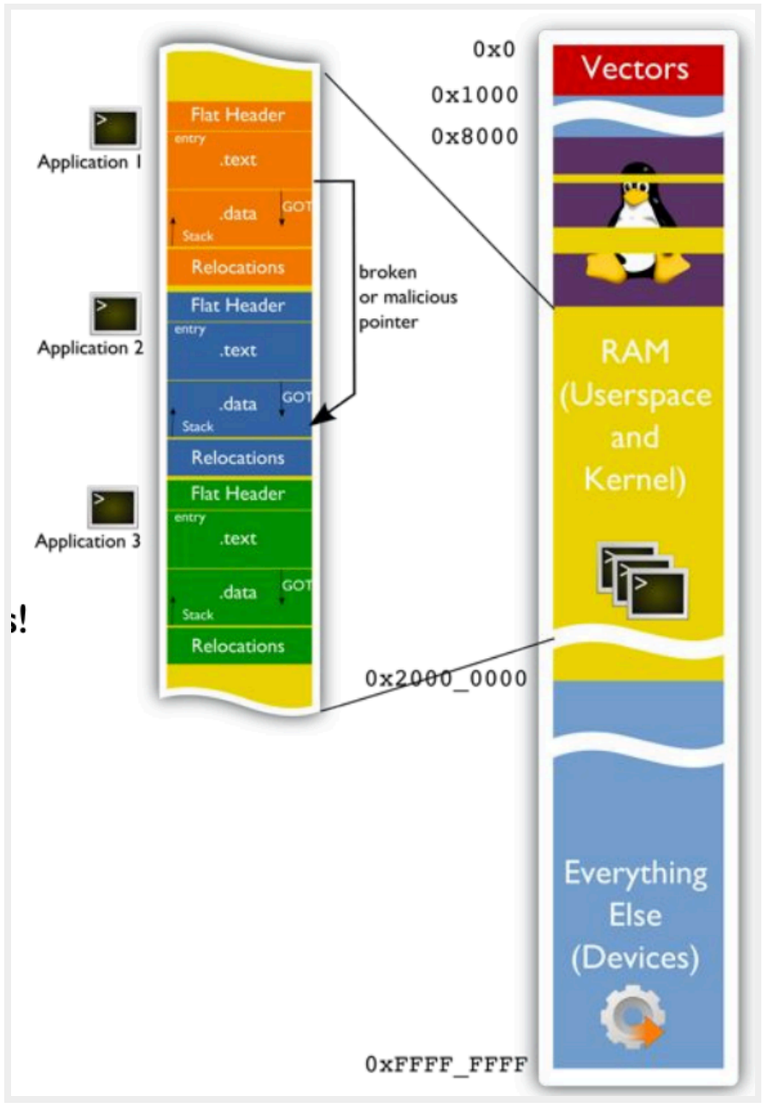
How to implement userspace network stack



- 1) protocol implementation (ARP, ICMP, TCP, etc)
- 2) device I/O interface (AF_XDP, DPDK, netmap, etc)
- 3) user interface (API, socket, others)
  - 1) can be implemented from scratch,
    but can be ported existing kernel (Linux/NetBSD)

port nommu kernel for a userspace benefits performance/feature-richness

nommu kernel



- mainly for embedded devices
- build w/ `CONFIG_MMU=n`
- features (restrictions)
  - physical == virtual address
  - no page table
  - no memory protection
- merged in linus tree since 2002

*ref: http://events17.linuxfoundation.org/sites/events/files/slides/uClinux%20ELC_43_small.pdf

nommu is nowadays considered as hard-to-maintain in Linux

Call for nommu LTP maintainer [was: Re: [PATCH 00/36] Remove UCLINUX from LTP]

Petr Vorel pvorel at suse.cz
Fri Jan 5 05:11:35 PST 2024

- Previous message (by thread): [PATCH 00/36] Remove UCLINUX from LTP
- Next message (by thread): Call for nommu LTP maintainer [was: Re: [PATCH 00/36] Remove UCLINUX from LTP]
- **Messages sorted by:** [date] [thread] [subject] [author]

```
Hi all,

[ Cc also automated-testing and buildroot ML
FYI thread started here:
https://lore.kernel.org/ltp/20240103015240.1065284-1-pvorel@suse.cz/ ]

> On 1/3/24 06:09, Cyril Hrubis wrote:
> > Hi!
> >> I am not sure I agree with this series.
> >> Removing support for UCLINUX from LTP is almost a guarantee for
> >> not noticing when more breakage is introduced.

> >> How exactly is UCLINUX broken in LTP?

> > As far as we know noone is using it and nobody is maintaing it for a
> > decade
```

- LTP (Linux test project) dropped nommu (uCLinux) support
- kernel often drops old architectures
  - no maintainers
  - increasing the number of bug reports
- nommu kernel was also discussed to drop
  - existing architectures (riscv/m68k/xtensa) is not agreeing

*ref: https://lore.kernel.org/ltp/20240105131135.GA1484621@pevik/

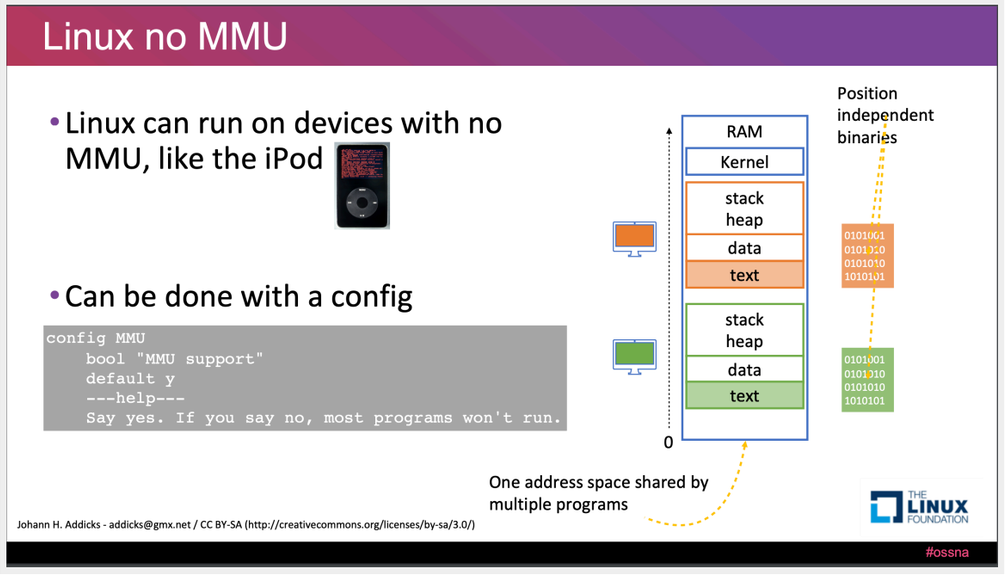
this is pity...

- still useful for small devices, FPGAs
- **nommu with virtualization**
  - if a guest doesn't need identical view with host
  - e.g, single process only guest (container/micro-service)
  - e.g., large number of instances, but needs genuine TCP implementation
- but lack of test infrastructure,
  - thus, hard to maintain ! (**<= now**)

implementation

nommu UML

- an extension to UML
  - `CONFIG_MMU=n`
  - different syscall entrypoint
    - **single address spaces** (processes/kernel)
    - run within a single host process (different from MMU-full UML)
  - prototype by Ricardo Koller (*1)
- patch (v14): https://lore.kernel.org/all/cover.1770170302.git.thehajime@gmail.com/




```
2025-11-08  8:05 Hajime Tazaki [this message]
2025-11-08  8:05 ` [PATCH v13 01/13] x86/um: nommu: elf loader for fdpic Hajime Tazaki
2025-11-08  8:05 ` [PATCH v13 02/13] um: decouple MMU specific code from the common part Hajime Tazaki
2025-11-08  8:05 ` [PATCH v13 03/13] um: nommu: memory handling Hajime Tazaki
2025-11-08  8:05 ` [PATCH v13 04/13] x86/um: nommu: syscall handling Hajime Tazaki
2025-11-08  8:05 ` [PATCH v13 05/13] um: nommu: seccomp syscalls hook Hajime Tazaki
2025-11-08  8:05 ` [PATCH v13 06/13] x86/um: nommu: process/thread handling Hajime Tazaki
2025-11-08  8:05 ` [PATCH v13 07/13] um: nommu: configure fs register on host syscall invocation Hajime Tazaki
2025-11-08  8:05 ` [PATCH v13 08/13] x86/um/vdso: nommu: vdso memory update Hajime Tazaki
2025-11-08  8:05 ` [PATCH v13 09/13] x86/um: nommu: signal handling Hajime Tazaki
2025-11-08  8:05 ` [PATCH v13 10/13] um: change machine name for uname output Hajime Tazaki
2025-11-08  8:05 ` [PATCH v13 11/13] um: nommu: disable SMP on nommu UML Hajime Tazaki
2025-11-08  8:05 ` [PATCH v13 12/13] um: nommu: add documentation of '' Hajime Tazaki
2025-11-08  8:05 ` [PATCH v13 13/13] um: nommu: plug nommu code into build system Hajime Tazaki
```

*1 Ricardo Koller, The most lightweight Virtual Machine Monitor is no Monitor at all, OSS-NA 2020.

nommu UML (cont'd)

- unlike LKL (Linux Kernel Library)
  - fully ABI compatible
  - multiple processes (e.g., shell)
  - arch dependent part is (slightly) large
    - runs on x86_64
- userspace: alpine linux + musl-libc/busybox nommu build
- features
  - fast: syscall => function call
  - existing UML facilities (e.g., KUnit, drivers)
  - KASAN (no NOMMU support before)
  - valgrind (experimental)
- limitations
  - generic nommu limitation (mmap, fork, etc)

Limitations

- **vfork(2)** instead of fork(2)
  - no CoW (copy-on-write)
  - parent blocks until child `exit(2)`
  - applications should be aware of
- **PIE** (position independent executable) binaries
  - objects w/o absolute address to execute in different location
- processes share the address space among others
  - no memory protection
- **mmap(2)**: a subset of functionalities
  - e.g., no `MAP_FIXED`, partial `MAP_SHARED` support

run as/in containers

```
% docker run -it -v /dev/shm:/dev/shm --rm  ghcr.io/thehajime/alpine:3.20.3-um-nommu
Core dump limits :
        soft - NONE
        hard - NONE
Checking environment variables for a tempdir...none found
Checking if /dev/shm is on tmpfs...OK
Checking PROT_EXEC mmap in /dev/shm...OK
Checking FSGSBASE instructions...OK
Adding 667009024 bytes to physical memory to account for exec-shield gap
Physical memory size shrunk to 2650800127 bytes
seccomp: setup filter syscalls in the range: 0x88c00000-0xfdffffff
(none):/# ls
bench.sh  dev       home      lmbench2  mnt       proc      run       srv       tmp       var
bin       etc       lib       media     opt       root      sbin      sys       usr
(none):/# uname -a
```

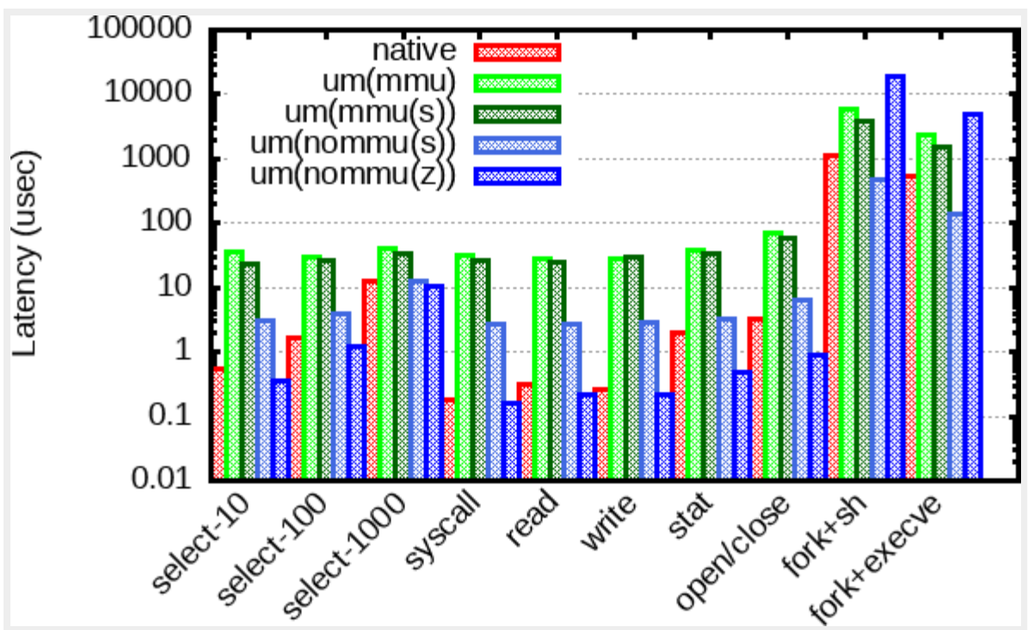
showcases

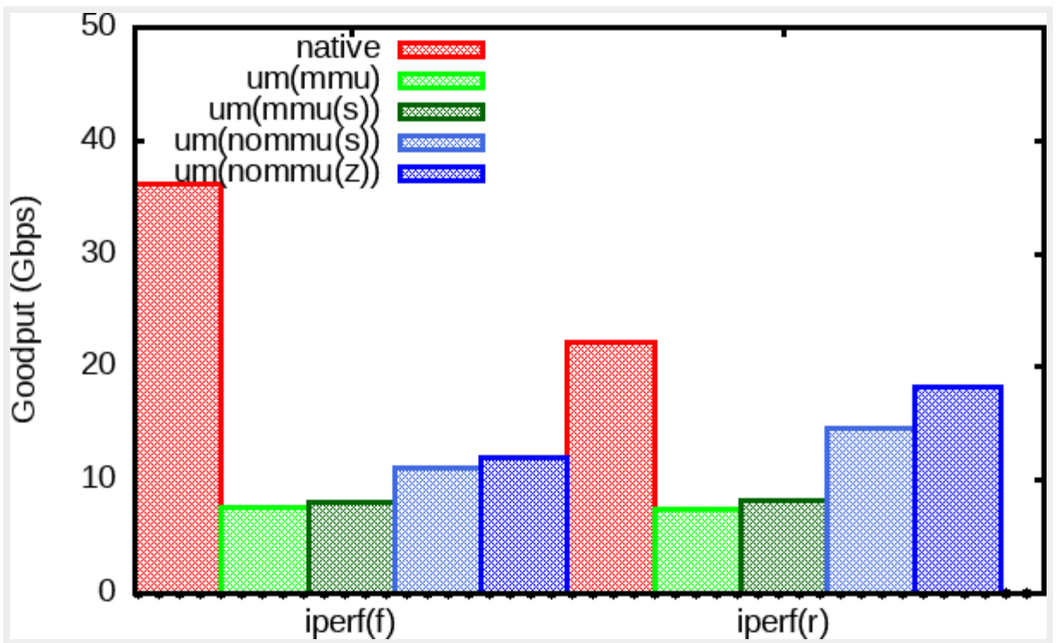
benchmarks: getpid(2)

	native	um	um-mmu(s)	um-nommu(s)	um-nommu(z)
getpid	173	33672	26940	2676	163

- 5 environments
  - `native` baremetal Linux
  - `um` UML with ptrace syscall hook
  - `um-mmu(s)` UML with seccomp syscall hook
  - `um-nommu(s)` UML + nommu with seccomp syscall hook
  - `um-nommu(z)` UML + nommu with zpoline syscall hook (experimental)

benchmarks: lmbench, iperf3





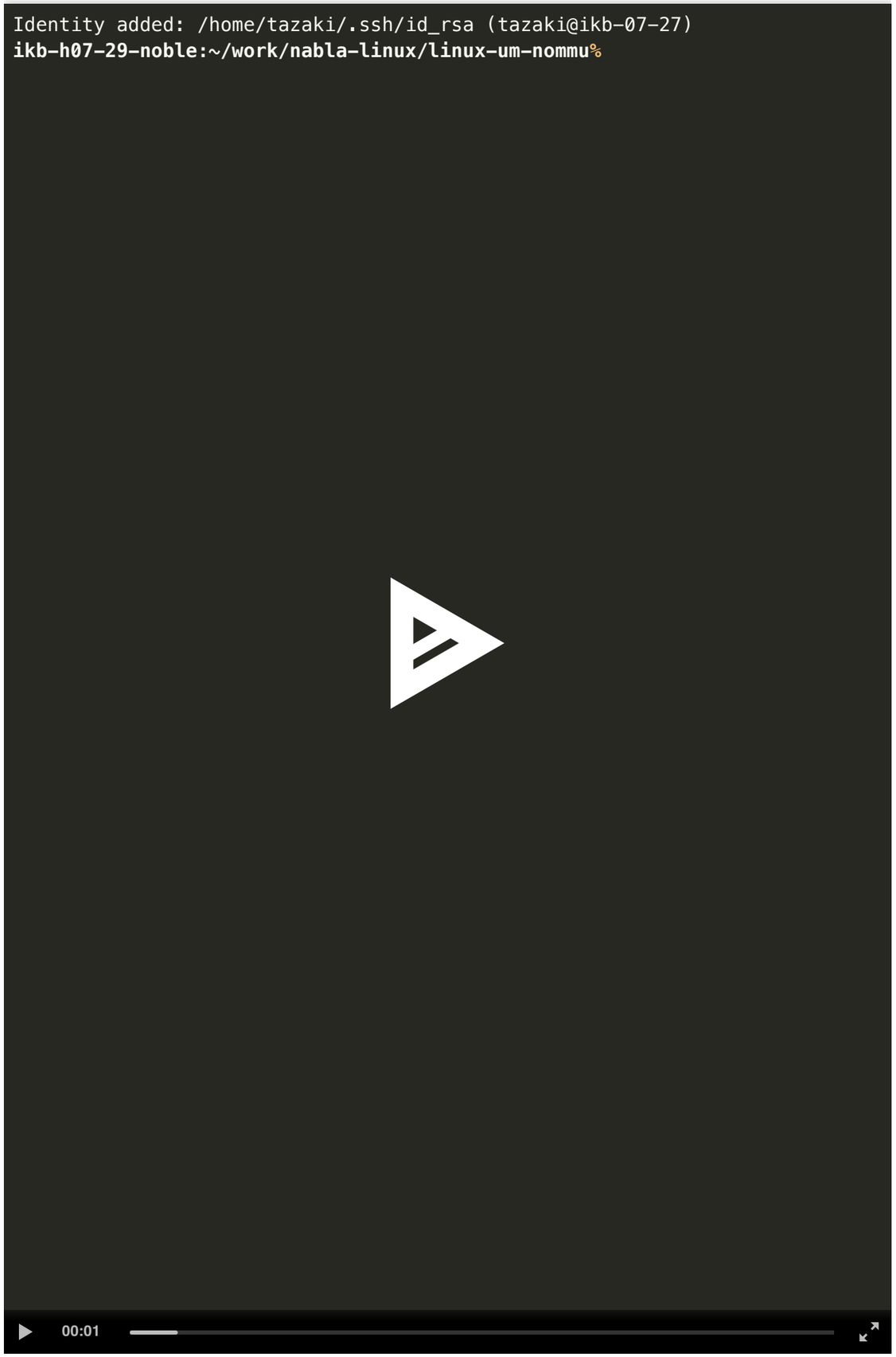
lmbench **iperf3**

- lmbench (lower is better)
- iperf3 (higher is better)
  - communicate with UML and host via a tap device
- findings
  - nommu: syscall is func call (fast)
  - ptrace (mmu) < seccomp(mmu) < seccomp(nommu)
  - TSO w/ UML (both mmu/nommu) doesn't fill the pipe
  - lmbench: fork+x with nommu(z) has slow (binary translation w/o cache)

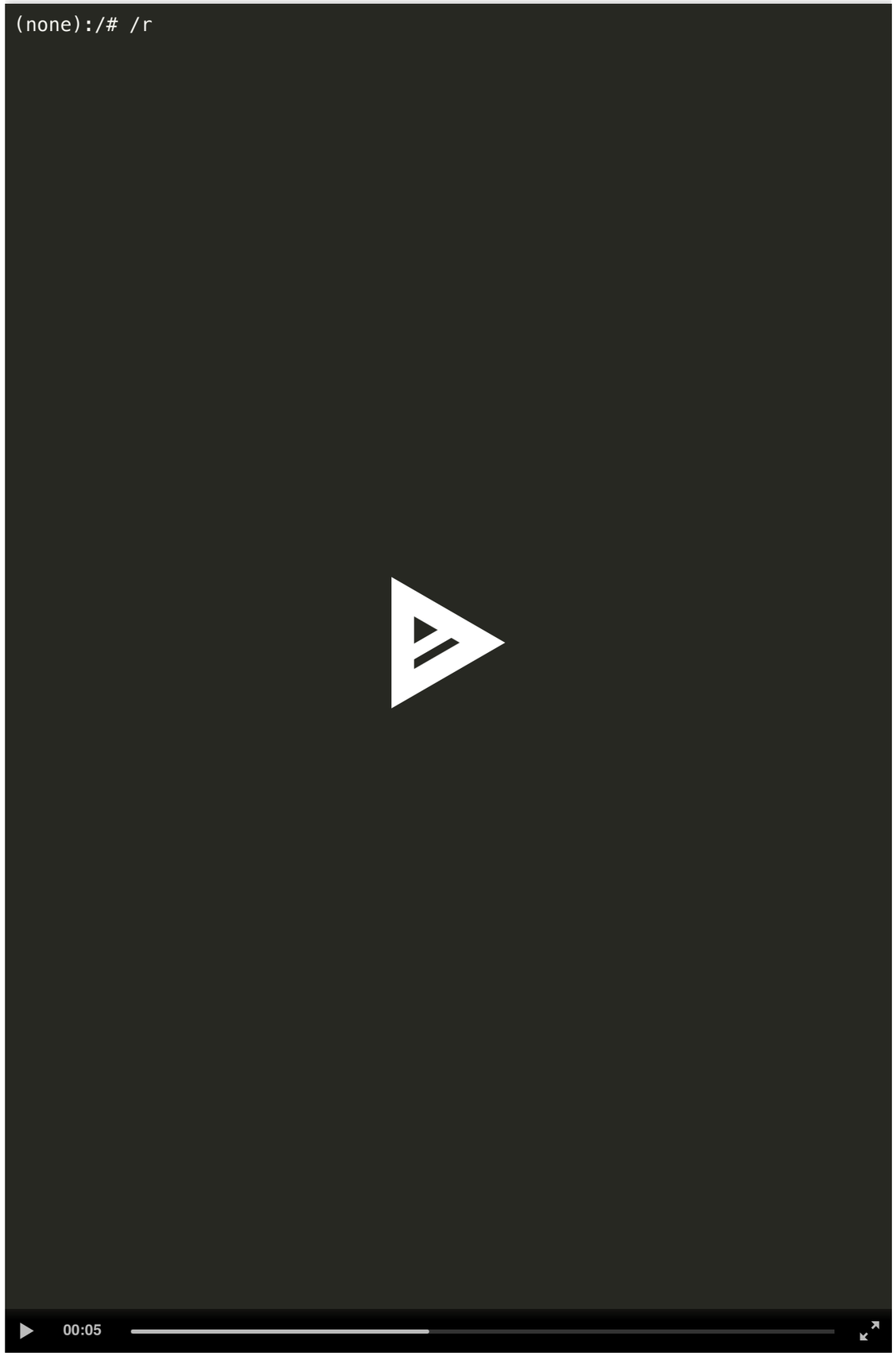
debugging/testing usecases

- detection nommu code **regression**
  - during maple tree refactor
  - https://lore.kernel.org/linux-mm/20241108222834.3625217-1-thehajime@gmail.com/
- **deadlock** on OOM
  - https://lore.kernel.org/all/m2fr96wjtw.wl-thehajime@gmail.com/

```
Modules linked in:
Pid: 38, comm: apk Not tainted 7.1.0-rc1-00044-g6d737c3b4548-dirty
RIP: 0033:acct_collect+0xa1/0x17e
RSP: 0000000070affe60  EFLAGS: 00010202
RAX: 000000000000000e RBX: 0000000000000000 RCX: 0000000070affe78
RDX: 0000000000000000 RSI: 0000000000000001 RDI: 00000000709b6200
RBP: 0000000070afff00 R08: 00000000709b6200 R09: 0000000000000001
R10: 00000000709b6208 R11: 00000000709b620c R12: 00000000605c9968
R13: 0000000000000001 R14: 00000000703fd400 R15: 000000006085a4c8
Kernel panic - not syncing: Kernel mode fault at addr 0x16, ip 0x600b710b
CPU: 0 UID: 0 PID: 38 Comm: apk Not tainted 7.1.0-rc1-00044-g6d737c3b4548-dirty #580 VOLUNTARY
Stack:
 00000000 704ba408 00000001 704ba3c8
 705df000 705defff 709b620c 00000000
```



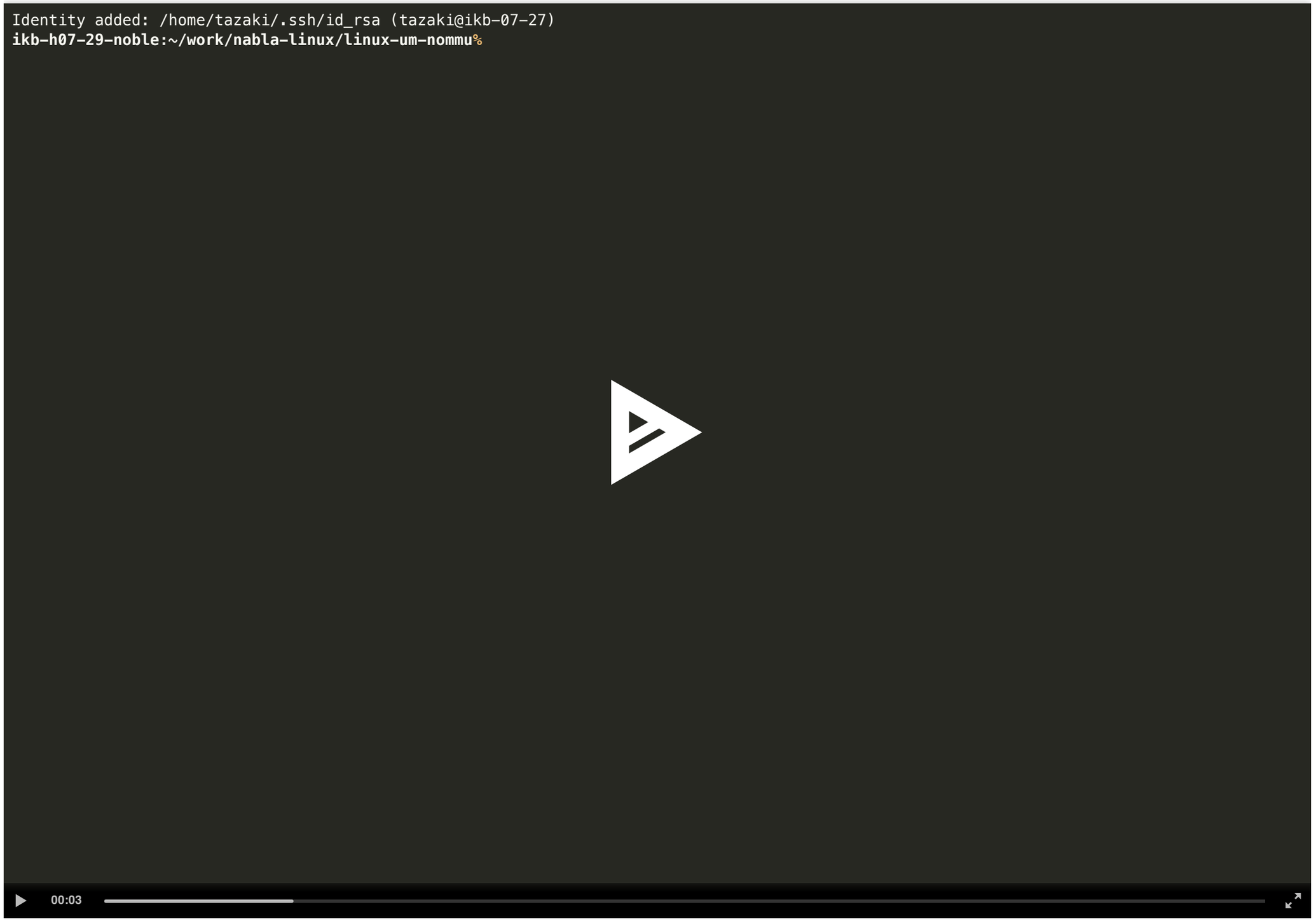
deadlock



deadlock fixed

KUnit test

```
./tools/testing/kunit/kunit.py run --kconfig_add CONFIG_MMU=n
```



case studies: a LTP test session

`testcases/kernel/syscalls/mseal/mseal02.c`

```
    void *addr;
    size_t pagesize = getpagesize();

    addr = mmap(NULL, pagesize * 4, PROT_READ | PROT_WRITE,
                    MAP_ANONYMOUS | MAP_PRIVATE, -1, 0);
    munmap(addr + pagesize * 1, pagesize);
```

```
(none):/# /root/shrink-mm
------------[ cut here ]------------
WARNING: include/linux/maple_tree.h:791 at __mas_set_range+0x97/0xb6, CPU#0: shrink-mm/37
Modules linked in:
CPU: 0 UID: 0 PID: 37 Comm: shrink-mm Not tainted 7.1.0-rc1-00045-ga98296e6ddcf-dirty #567 VOLUNTARY
Stack:
 7089fcc0 607f7c7d 605bfca9 00000001
 ffffff00 607f7c7d 600233ce 00000317
 7089fcf0 6002f249 00000000 00000000
Call Trace:
 [<600233ce>] ? _printk+0x0/0x5b
 [<60031fec>] show_stack+0x11c/0x12b
 [<605bfca9>] ? dump_stack_print_info+0x0/0x12f
 [<600233ce>] ? _printk+0x0/0x5b
 [<6002f249>] dump_stack_lvl+0x65/0x80
```

- if `munmap()` partial region, causing `vma_split()`, clearing memory becomes puzzled due to an odd pointer of the tree iterator
- patch proposed to mm-tree

*ref: https://github.com/linux-test-project/ltp/blob/master/testcases/kernel/syscalls/mseal/mseal02.c

Testing LTP

LTP Results

Environment information	
distribution	alpine
distribution_version	3.20.3
kernel	Linux 7.1.0-rc1-g6d737c3b4548-dirty #1 Wed Jul 8 06:04:14 UTC 2026
cmdline	vec0:transport=raw,ifname=eth0,depth=128,gro=1 root=/dev/root rootflags=/ rootfstype=hostfs rw loglevel=8 zpoline=0 console=tty init=/ltp-exec.sh LTP_SINGLE_FS_TYPE=ext4
arch	um(nommu)/x86_64
cpu	unknown
swap	0 kB
RAM	2536432 kB

Overall results					
Runtime: 0:11:17.604661	Passed: 6773	Skipped: 1049	Failed: 76	Broken: 4	Warnings: 5

Hide Passed Hide Skipped Filter by ID:

Test ID ↕	Duration ↕	Passes ↕	Skips ↕	Fails ↕	Broken ↕	Warns ↕
accept01	0.00	5	0	0	0	0
accept02	0.00	1	0	0	0	0
accept03	0.01	18	5	0	0	0
accept4_01	0.01	8	1	0	0	0
access01	0.01	199	0	0	0	0
access02	0.01	16	0	0	0	0
access03	0.01	0	8	0	0	0
access04	0.00	12	0	0	0	0
acct01	0.01	10	1	0	0	0
acct02	1.01	1	0	0	0	0
add_key01	0.00	6	2	0	0	0
add_key02	0.00	0	9	0	0	0
add_key03	0.00	1	0	0	0	0
add_key04	0.00	1	0	0	0	0
add_key05	0.00	0	1	0	0	0
adjtimex01	0.00	2	0	0	0	0

next steps

- more studies w/ tests
  - implement selftests for mm/nommu
  - KASAN CI
  - syzbot
  - stabilize LTP results
  - automations (nightly, per-commit)

summary

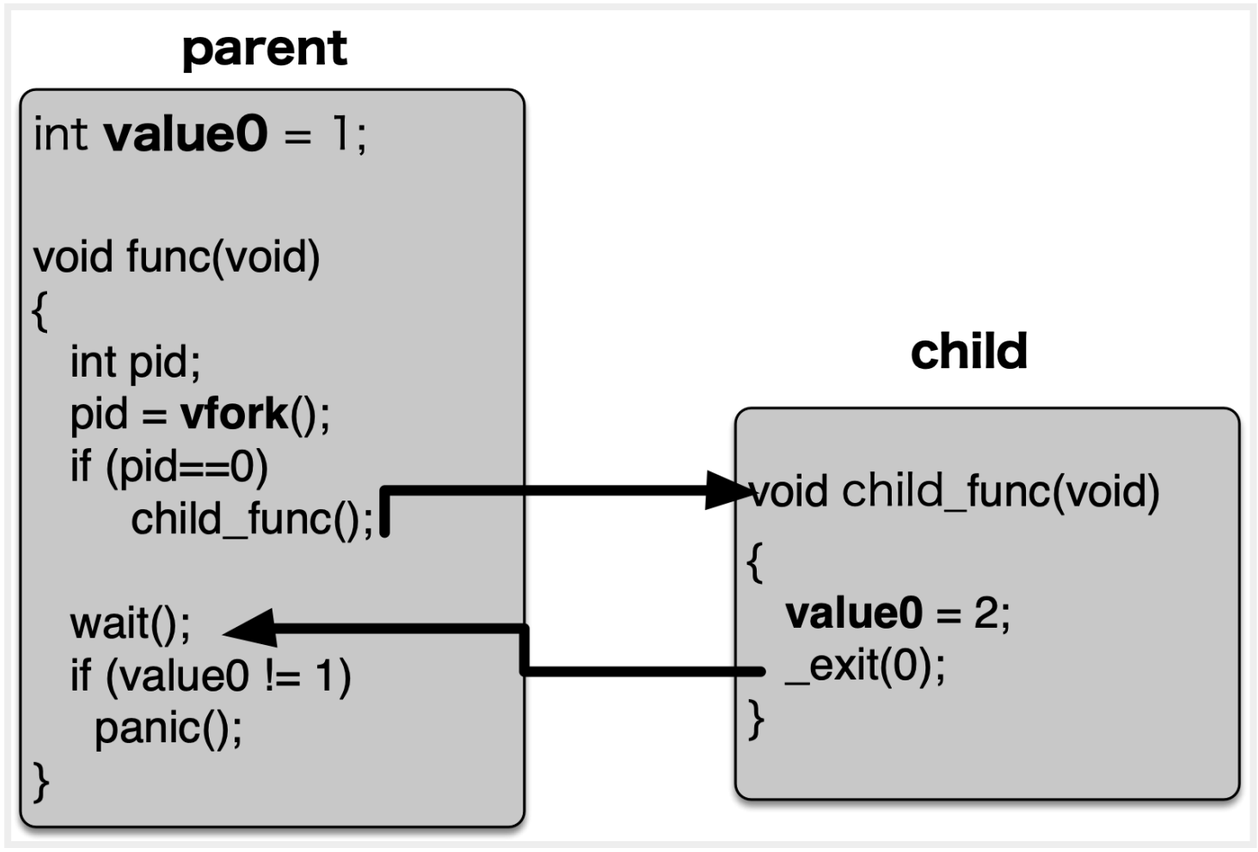
- nommu has a usecase for virtualization
  - **essence from userspace network stacks**
- add a new nommu port (to Linux tree) to help testing/debugging

current UML patch (v14):
https://lore.kernel.org/all/cover.1770170302.git.thehajime@gmail.com/

Backups

Limitations: vfork(2)

- basics of vfork
  - parent process blocks until child calls {_exit()/exec()}
  - share memory between parent/children (**including global symbols**)
  - you cannot parallel processes for multiple incoming requests (e.g., nginx worker)



LTP results

	passed	failed	broken	skipped	warnings
native	9850	81	24	241	0
mmu	9063	62	23	515	3
mmu-s	9060	65	23	514	3
nommu-s	6761	65	4	1049	5
nommu-z	6759	64	4	1049	5
riscv-nommu	4068	135	52	654	6

Why fast ? context switch: uml vs nommu-um

```
    struct timespec ts1, ts2;
    clock_gettime(CLOCK_REALTIME, &ts1);  // 1)
    getpid()                              // 2)
    clock_gettime(CLOCK_REALTIME, &ts2);  // 3)
```

- seccomp-mmu, 2)-3)= 11 usec

```
=> 1)   uml-userspace-3092637 [002] 1749286.670199: signal_generate:      sig=31 errno=0 code=1 comm=uml-us
 |      uml-userspace-3092637 [002] 1749286.670200: sys_enter_futex:      op=FUTEX_WAKE uaddr=0x7fffffffdf8
 |      uml-userspace-3092637 [002] 1749286.670201: sys_enter_futex:      op=FUTEX_WAIT uaddr=0x7fffffffdf8
 |      uml-userspace-3092637 [002] 1749286.670202: sched_switch:         uml-userspace:3092637 [120] S ==>
 |            <idle>-0       [028] 1749286.670203: sched_switch:         swapper/28:0 [120] R ==> vmlinux:
12 usec       vmlinux-3092631 [028] 1749286.670205: sys_enter_futex:      op=FUTEX_WAKE uaddr=0x60b64f8c va
 |            vmlinux-3092631 [028] 1749286.670206: sys_enter_futex:      op=FUTEX_WAIT uaddr=0x60b64f8c va
 |            vmlinux-3092631 [028] 1749286.670207: sched_switch:         vmlinux:3092631 [120] S ==> swapp
 |               <idle>-0     [002] 1749286.670209: sched_switch:         swapper/2:0 [120] R ==> uml-users
=> 2)   uml-userspace-3092637 [002] 1749286.670211: signal_generate:      sig=31 errno=0 code=1 comm=uml-us
 |      uml-userspace-3092637 [002] 1749286.670212: sys_enter_futex:      op=FUTEX_WAKE uaddr=0x7fffffffdf8
 |      uml-userspace-3092637 [002] 1749286.670213: sys_enter_futex:      op=FUTEX_WAIT uaddr=0x7fffffffdf8
 |      uml-userspace-3092637 [002] 1749286.670214: sched_switch:         uml-userspace:3092637 [120] S ==>
11 usec         <idle>-0       [028] 1749286.670215: sched_switch:         swapper/28:0 [120] R ==> vmlinux:
 |            vmlinux-3092631 [028] 1749286.670216: sys_enter_futex:      op=FUTEX_WAKE uaddr=0x60b64f8c va
```

- sig=31 (SIGSYS) indicate the timing of syscall triggered
- between 2) and 3), two processes (uml-userspace/vmlinux) sync w/ futex
- finally switched, returns to userspace

Why fast ? (cont'd)

- seccomp-nommu, 2)-3) = 3 usec

```
1)        vmlinux-3092542 [006] 1749158.829292: signal_generate:      sig=31 errno=0 code=1 comm=vmlinux pid=
2) 2usec  vmlinux-3092542 [006] 1749158.829294: signal_generate:      sig=31 errno=0 code=1 comm=vmlinux pid=
3) 3usec  vmlinux-3092542 [006] 1749158.829297: signal_generate:      sig=31 errno=0 code=1 comm=vmlinux pid=
```

- kernel memory space can exist in userspace, in single address space.
- efficient, but may be danger.

Discussions

- You can do what (original) UML can
- If virtualization is used
  - for doing the same in real
    - software development, testing,
      => qemu/kvm is yours
  - for doing similar but more small jobs
    - e.g., run a process in a guest VM
    - => vm with nommu kernel is an option