# Address Space Isolation (ASI)

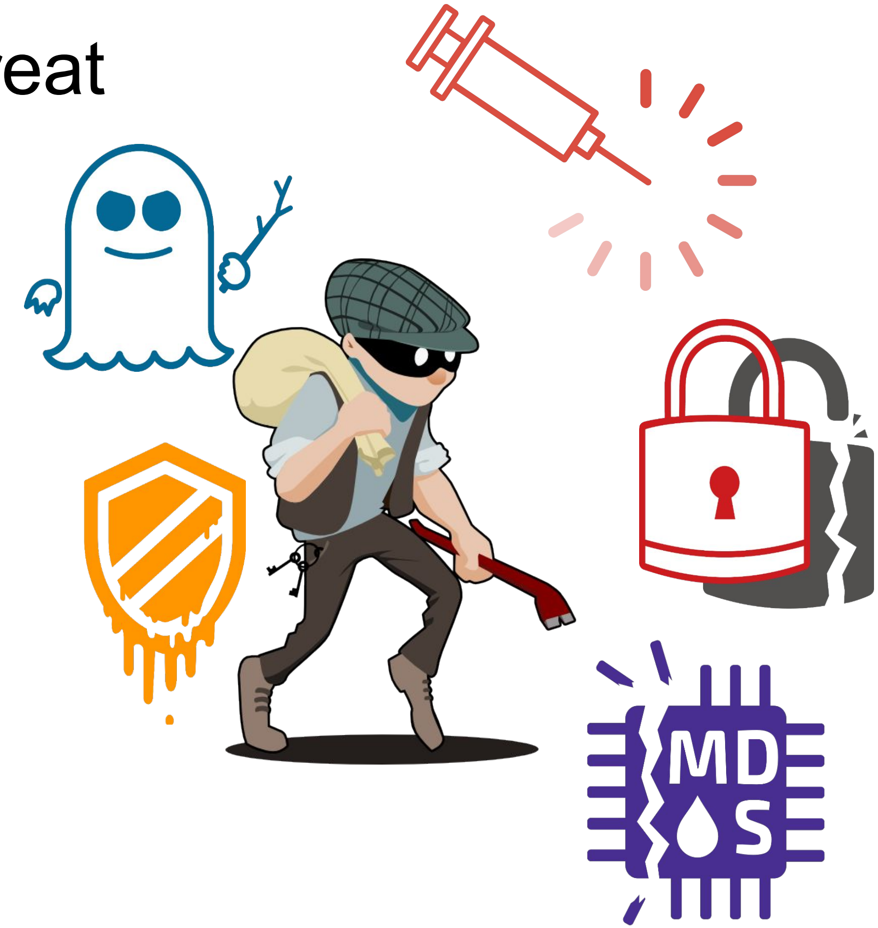
Speculative execution protection

Google

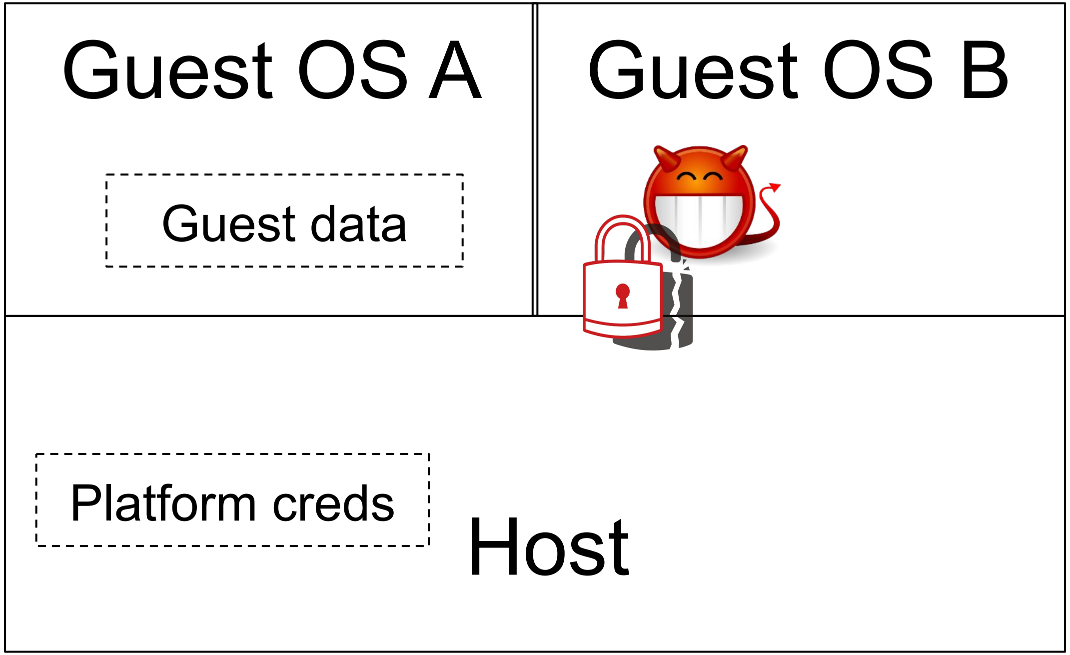
Ofir Weisse, Junaid Shahid, Oleg Rombakh, and Paul Turner

# The Speculative Attacks Threat

- These are µ-architectural attacks
- They break architectural boundaries
  - User/kernel boundary
  - Inter-process boundary
  - **<u>VM/host boundary</u>**
- They therefore compromise
  - Our customer’s data
  - Infrastructure (host) credentials
- Current mitigations are either
  - High overhead, or
  - Incomplete



# What Can be Stolen



# Roadmap

- The Speculative Attacks Threat
- **L1TF Refresher**
- Why Mitigation is Challenging
- Address Space Isolation (ASI)

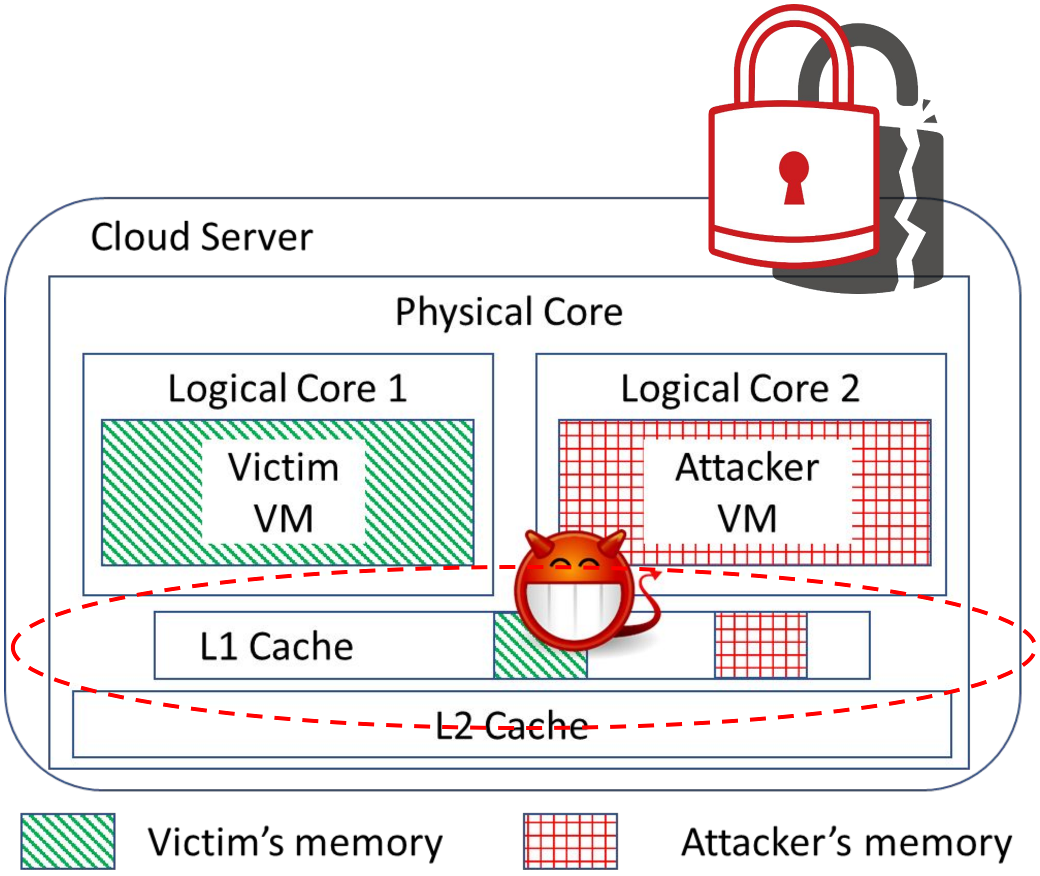
To learn more about speculative attacks:

foreshadowattack.eu

ofirweisse.com/MICRO2019_NDA.pdf

# L1TF in a Nutshell

- Shared µ-arch state can be stolen
  - L1TF - L1 cache
  - MDS - other µ-buffers
- The state can be left by previous context
- Or provoked by the attacker
  - Via calling an API



# Roadmap

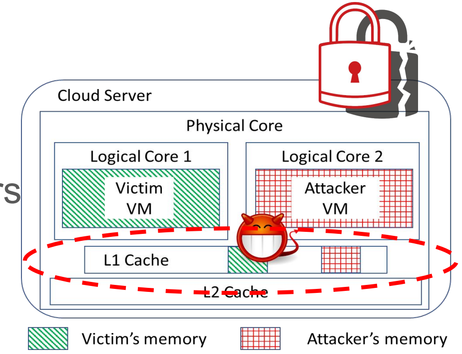
- The Speculative Attacks Threat
- L1TF Refresher
- **Why Mitigation is Challenging**
- Address Space Isolation (ASI)
- Initial Results

# The Challenge: Mitigations are Hard

1. Stop speculation, e.g., with lfences everywhere
   - X Extremely slow
2. Stop side-channels - that's a cat and mouse came
   - X E.g., L1D-cache, L1I-cache, BTB, branch-direction-predictor, etc. etc.
3. Stop speculation after branches
   - X Slow
   - X Error-prone
   - X Doesn't stop L1TF, MDS, etc

# The Challenge: Mitigations are Hard

1. Stop speculation, e.g.,  with lfences everywhere
   - X Extremely slow
2. Stop side-channels - that’s a cat and mouse came
   - X E.g., L1D-cache, L1I-cache, BTB, branch-direction-predictor, etc. etc.
3. Stop speculation after branches
   - X Slow
   - X Error-prone
4. Scrub/flush secrets from state (L1 cache and other buffers)
   - X The attacker can trigger execution bringing data to these buffers
   - X The execution above can even be speculative!
   - X Async execution (interrupts), Hardware prefetch are additional vectors
5. HyperThreading complicates defenses event more!
   - X A sibling thread can snoop shared resources

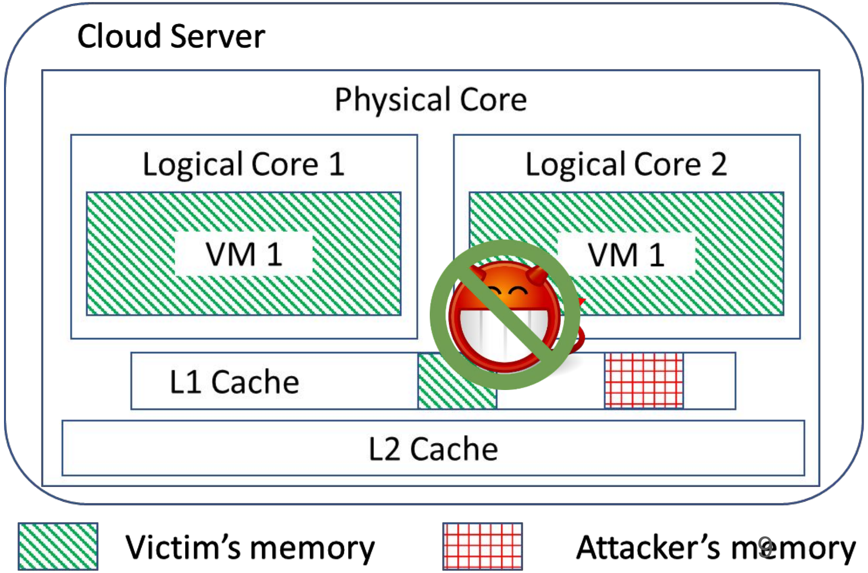


# What mitigations are applied today? (1)

Disabling HyperThreading infeasible (cost, performance, etc)

So what can we do?

- Secure core scheduling
  - Never run two VMs on the same physical core

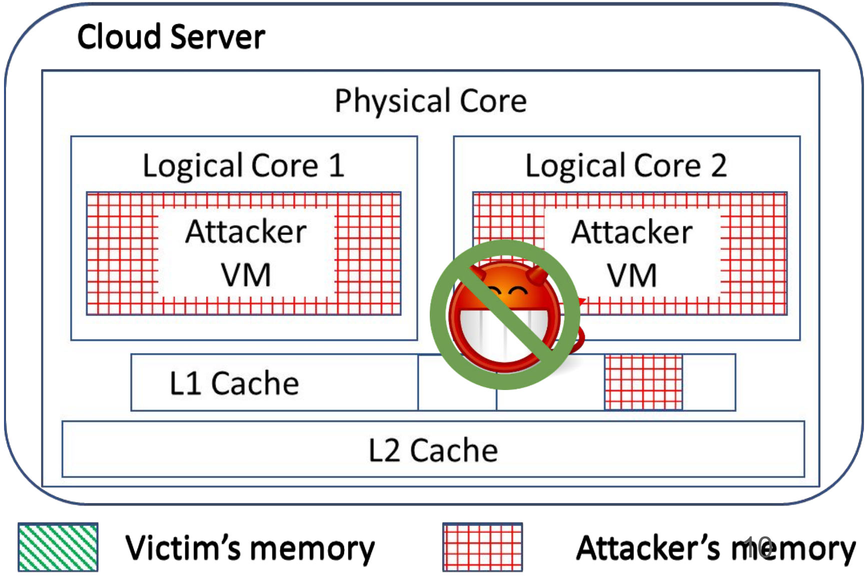



# What mitigations are applied today? (2)

Disabling HyperThreading is costly for performance/capacity

So what can we do?

- Secure core scheduling
- Flush L1 cache on VMENTER
  - Expensive

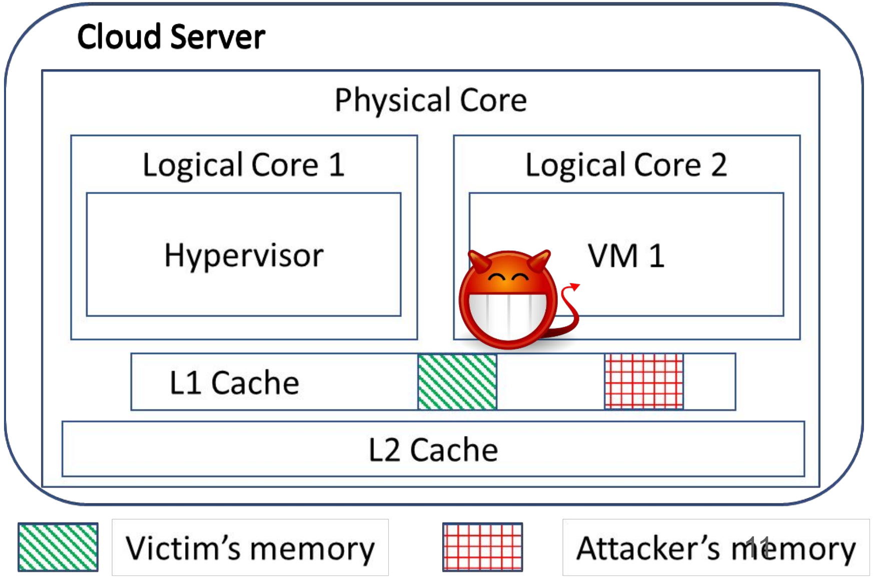


# What mitigations are applied today? (3)

Disabling HyperThreading is devastating for performance

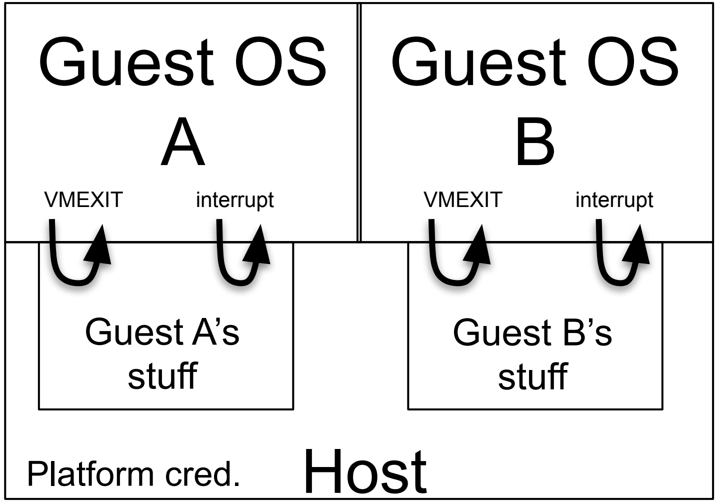
So what can we do?

- Secure core scheduling
- Flush L1 cache on VMENTER
- On VMEXIT to hypervisor – make sure other sibling core is stunned (not running)
    - Very expensive



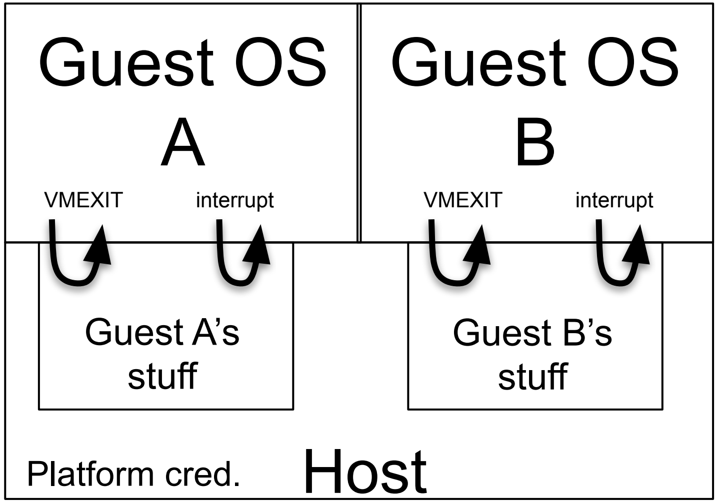
# What attack surface is open w/o constant flushes?

- On VMEXIT, interrupt handling may bring into cache/uarch-buffers data that
  - Belongs to other guests or
  - Is a platform secret
- That data can later be stolen via, e.g., L1TF
  - By the VM running after VMENTER
  - By sibling core during hypervisor execution

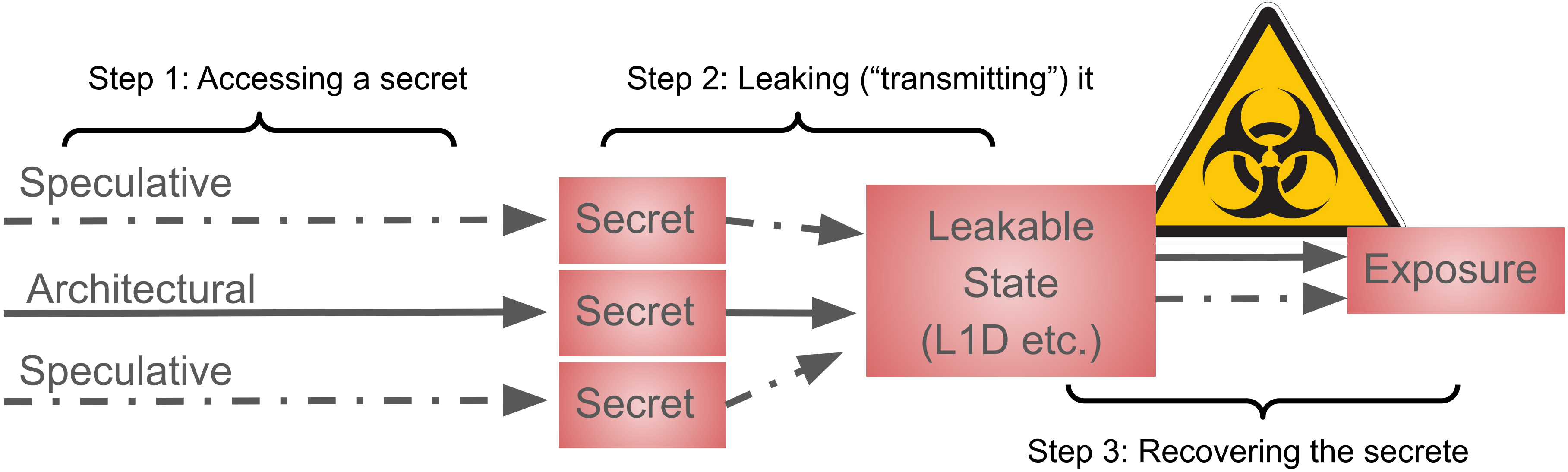


# What attack surface is open w/o constant flushes?

- On VMEXIT, interrupt handling may bring into cache/uarch-buffers data that
  - Belongs to other guests or
  - Is a platform secret
- That data can later be stolen via, e.g., L1TF
  - By the VM running after VMENTER
  - By sibling core during hypervisor execution
- Block-list approaches, i.e., removing specific sensitive memory, are may lead to a whac-a-mole



# Rethinking Mitigation - Understanding the Leak




**Status quo**: u-arch buffers are always (potentially) contaminated with secrets

**Sad conclusion:** Need to either a) stop speculation or b) continuously scrub state

For more details: ofirweisse.com/MICRO2019_NDA.pdf

# Rethinking Mitigation - Understanding the Leak

Step 1: Accessing a

Speculative

Architectural

Speculative

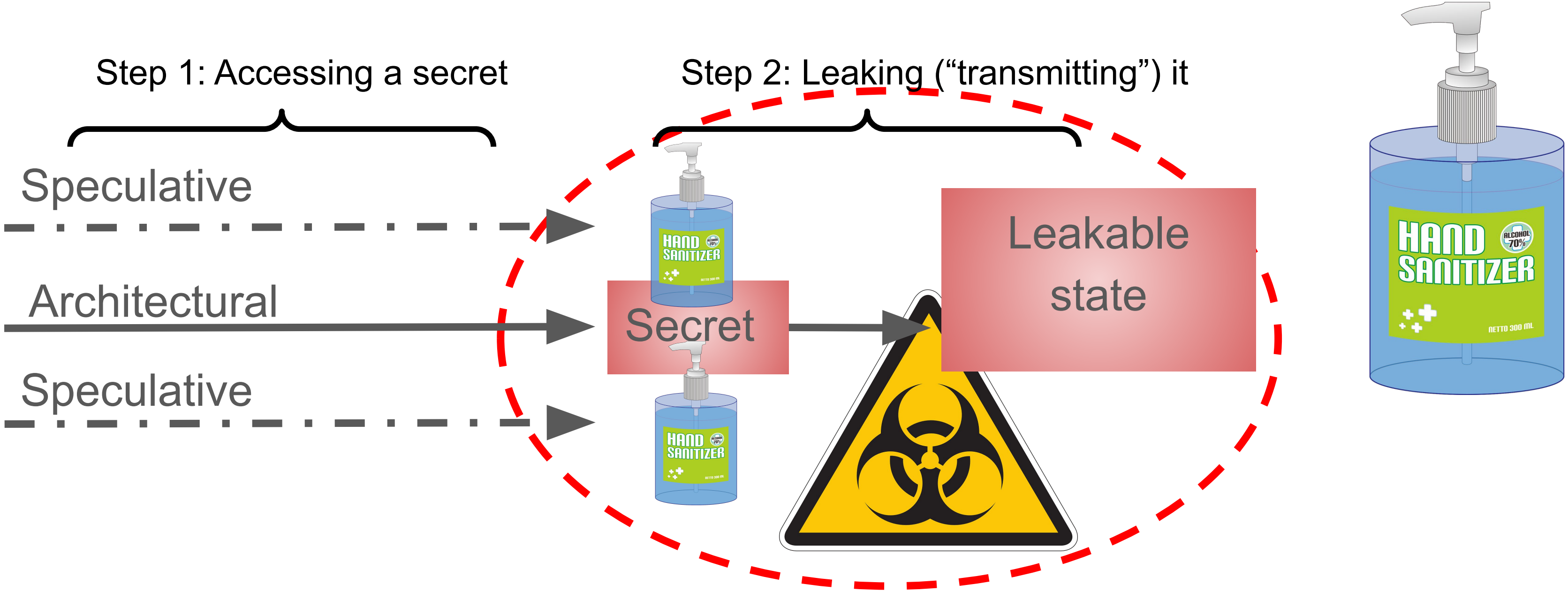

Exposure

overing the secrete

**Status quo**: u-arch buffers are always (potentially) contaminated with secrets

**Sad conclusion:** Need to either a) stop speculation or b) continuously scrub state

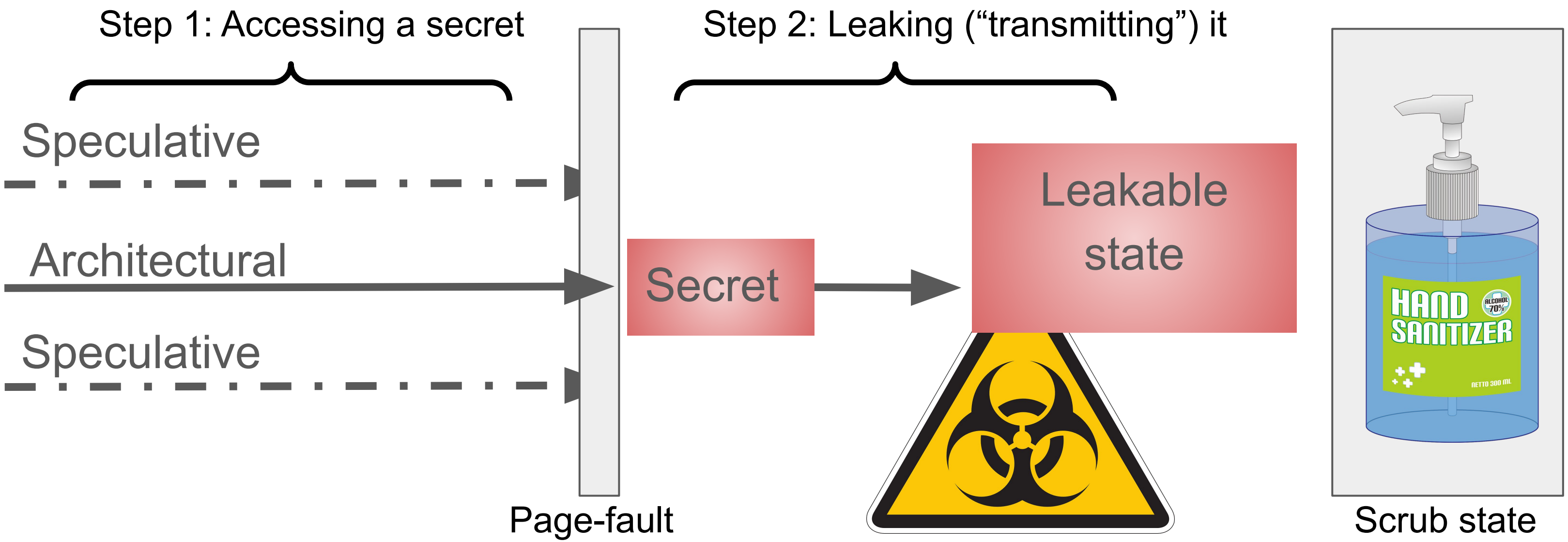
# Rethinking Mitigation - Limiting Exposure




**We want a way to circumscribe access to secrets and leakable state.**

**We then apply protection only when secrets are "in flight"**

# Idea: #PF as a fork between speculative & non-spec exec




**We want a way to circumscribe access to secrets and leakable state.**

**We then apply protection only when secrets are “in flight”**

# Trivial example: Spectre V1 (bounds check bypass)

```
int foo(u8 *arr, int size, int index) {
        if (index < size) {
                // Should lfence
                return global_array[ arr[index]* 64 ];
        }
        // ...
}
```

If index is out of bounds, "`arr`" might speculatively still be accessed.

# Trivial example: Spectre V1 (bounds check bypass)

```
int foo(u8 *arr, int size, int index) {
        if (index < size) {
                // Should lfence
                return global_array[ arr[index]* 64 ];
        }
        // ...
}
```

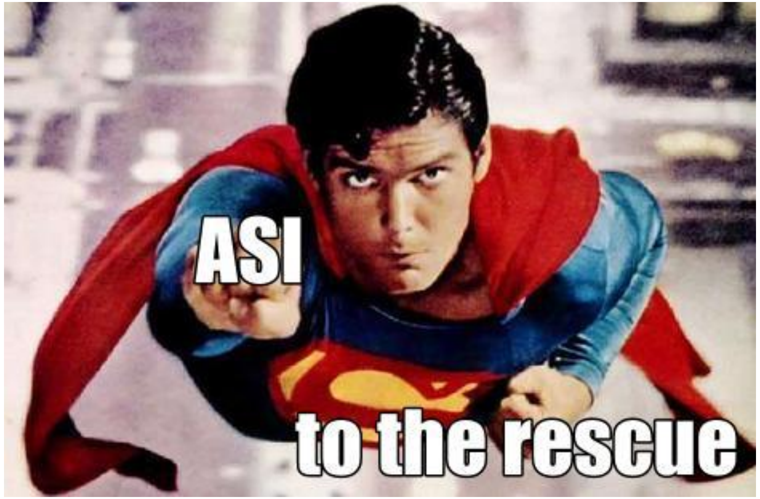
If index is out of bounds, “`arr`” might speculatively still be accessed.

If `&arr[index]` is not mapped in the page-table → page-fault

**Question:** When do we scrub clean??

# Roadmap

- The Speculative Attacks Threat
- L1TF Refresher
- Why Mitigation is Challenging
- **Address Space Isolation (ASI)**
- Initial Results



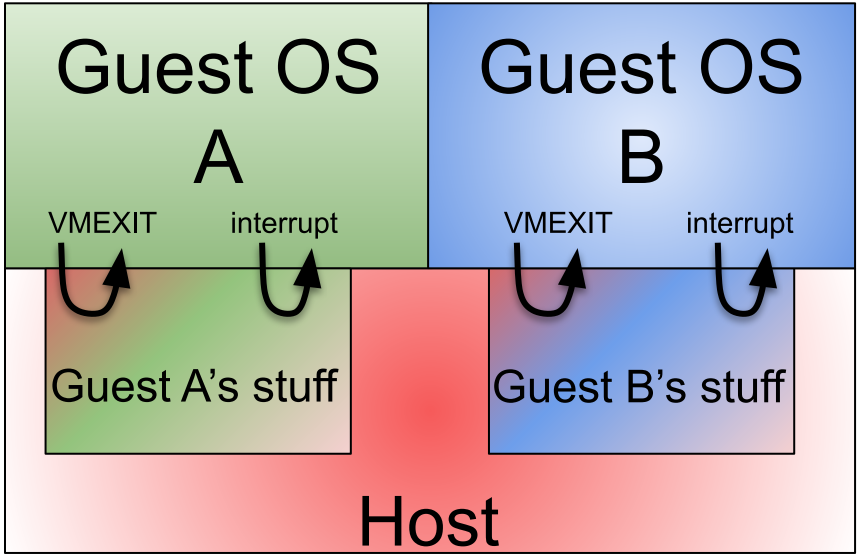
# Concurrent efforts

Eventually - we need a complete solution for Linux

- Intel - unmap guest memory from the direct map (KVM protected memory)
  - One VM cannot access memory of another VM
- IBM - protecting containers
  - Allocate namespace-private memory
  - Per-process private (userspace) memory
  - Remove mapping from the global page-table
- Oracle
  - KVM address space isolation, similar to our effort (e.g. #pf-fork)
- Amazon
  - Allocate process local memory, removed from the direct map.

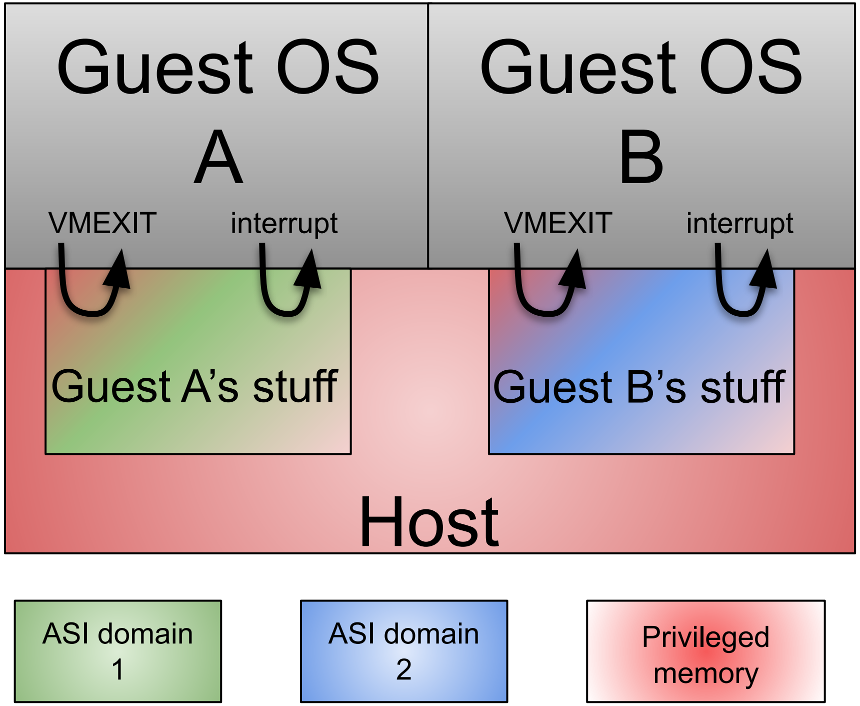
# Address Space Isolation - Premise

- On most VMEXIT's, the hypervisor only touches
  - Current guest stuff
  - Non sensitive data at the host



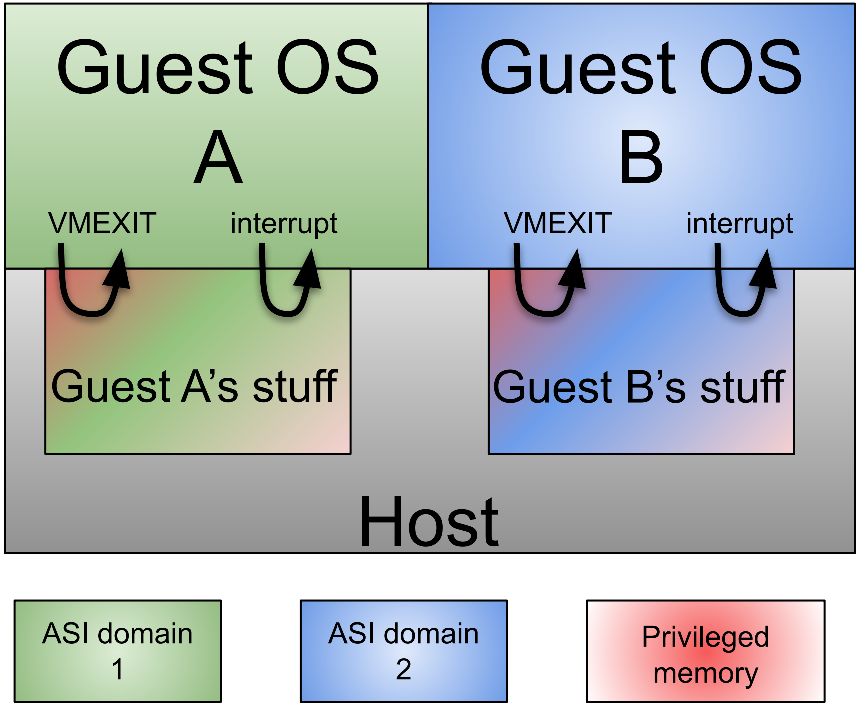
# Address Space Isolation - Basic Idea

- Split kernel memory to privileged and unprivileged-domains
- Each domain has a seperate page-table
- Touching data out of a domain results in a page-fault - <u>cannot be speculative</u>
- At first, only include kernel addresses



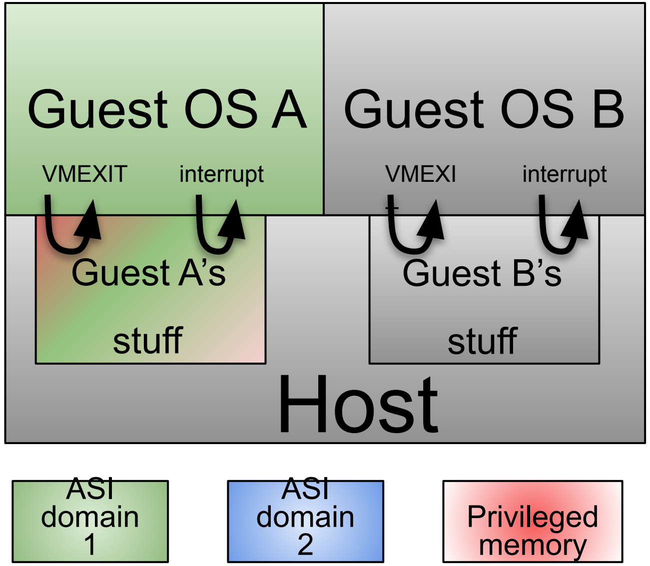
# Address Space Isolation - Basic Idea

- Split kernel memory to privileged and unprivileged-domains
- Each domain has a seperate page-table
- Touching data out of a domain results in a page-fault - <u>cannot be speculative</u>
- At first, only include kernel addresses
- ASI can be extended to include userspace memory



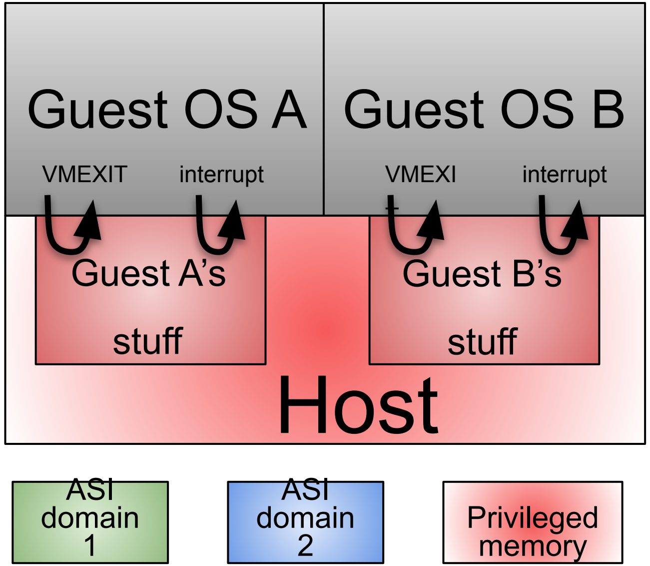
# ASI Lifecycle

```
//IOCTL KVM_RUN
for (;;) { // in vcpu_run()
    // call vmx_vcpu_run()
      asi_enter(); // Switch CR3 to unprivileged map
        // VMENTER
        // VMEXIT by the platform
        // Try to handle exit, may touch
           privileged data, which will cause
           A page fault --> asi_exit()
}
```



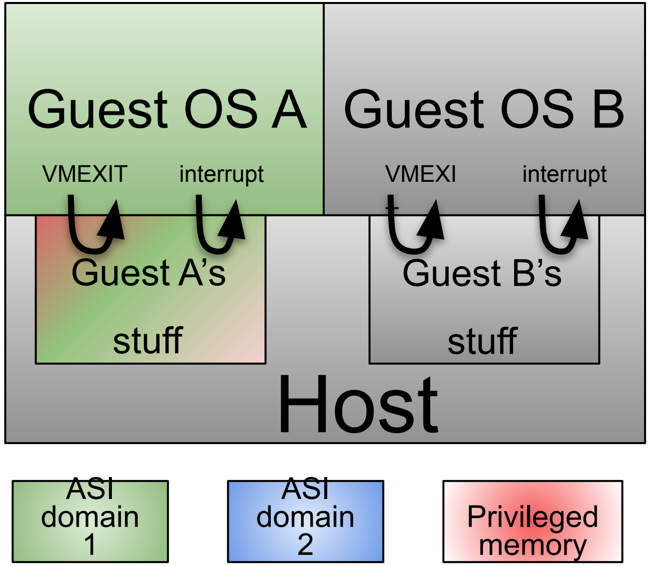
# What happens on a page-fault?

1. Call asi_exit() which will:
2. Call pre_asi_exit() callback which will
   a. Stun sibling core
   b. Log exit stat
3. Switch page table (CR3 in Intel) to the privileged page-table



# What happens on re-entry via asi_enter()?

1. Switch page table (CR3 in Intel) to the un-privileged Page-table
2. Call post_asi_enter() callback which will
    a. Flush L1D cache and any other uarch buffer
    b. Unstun sibling core

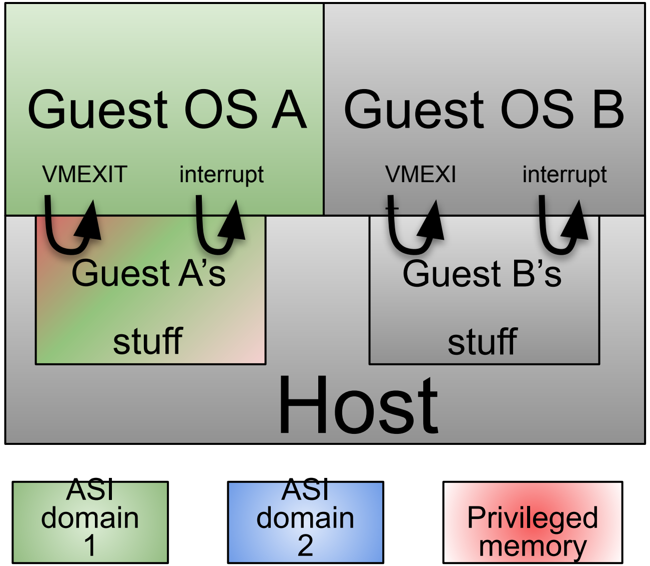
# Challenges

1. What data is OK to place within the unprivileged map?
   a. Anything that belongs to the guest anyhow
   b. Kernel maintenance structures which are used frequently and are not sensitive
2. How to handle PF/asi_exits within interrupts, nmi's, etc.?
   a. Must automatically re-asi_enter() when done


# Handling Interrupts

```
//IOCTL KVM_RUN
for (;;) { // in vcpu_run()
    // call vmx_vcpu_run()
        asi_enter(); // Switch CR3 to unprivileged map
         // Interrupt? We must re-enter!
         // VMENTER
         // VMEXIT by the platform
         // Interrupt? Meh..
         // Try to handle exit, may touch
             privileged data, which will cause
             A page fault --> asi_exit()

}
```

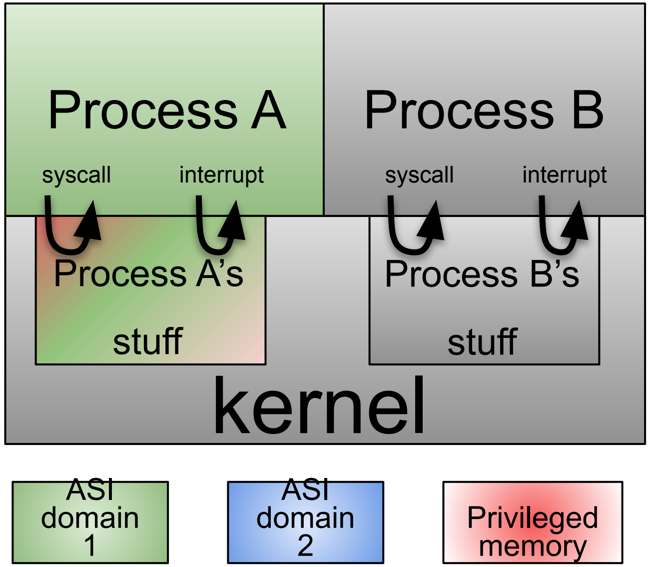



# Challenges

1. What data is OK to place within the unprivileged map?
    a. Anything that belongs to the guest anyhow
    b. Kernel maintenance structures which are used frequently and are not sensitive
2. How to handle PF/asi_exits within interrupts, nmi’s etc.?
    a. Must automatically re-asi_enter() when done
3. Integration with KPTI
    a. Eventually ASI will hopefully also replace KPTI. Both write to CR3.
4. How to manage dynamic allocations (kmalloc/vmalloc)?
    a. Some allocations are process specific, others are system-wide
    b. We want to avoid synchronization between page tables
    c. We want to minimize system wide tlb-flushes
5. In nested virtualization, L1 guest memory should be protected from L2

# ASI as a replacement for KPTI

- KPTI switches page-tables upon entry/exit to the kernel
- ASI (sometimes) switches page-tables upon entry/exit from a VM
- The same approach can, therefore, replace KPTI
    - To minimize page-table switches



# Initial Results - Redis YCSB

Ratio of ASI-exits/VM-exits

```
KVM/VCPU 0xffffc9001da89000/0: Time 309.05 seconds, asi/vm exits = 46160 / 4506402 = 1.02 %
KVM/VCPU 0xffffc9001da89000/1: Time 291.67 seconds, asi/vm exits = 400531 / 1267665 = 31.60 %
KVM/VCPU 0xffffc9001da89000/2: Time 291.67 seconds, asi/vm exits = 413946 / 2323131 = 17.82 %
KVM/VCPU 0xffffc9001da89000/3: Time 291.63 seconds, asi/vm exits = 499027 / 1045507 = 47.73 %
KVM/VCPU 0xffffc9001da89000/4: Time 291.69 seconds, asi/vm exits = 482687 / 2013058 = 23.98 %
KVM/VCPU 0xffffc9001da89000/5: Time 291.62 seconds, asi/vm exits = 500809 / 2170556 = 23.07 %
KVM/VCPU 0xffffc9001da89000/6: Time 291.68 seconds, asi/vm exits = 478710 / 1775451 = 26.96 %
KVM/VCPU 0xffffc9001da89000/7: Time 291.61 seconds, asi/vm exits = 482880 / 2059408 = 23.45 %
total_asi_exits = 3304750
KVM/VCPU 0xffffc90039f35000/0: Time 225.19 seconds, asi/vm exits = 489981 / 6257089 = 7.83 %
KVM/VCPU 0xffffc90039f35000/1: Time 225.00 seconds, asi/vm exits = 493745 / 1009584 = 48.91 %
KVM/VCPU 0xffffc90039f35000/2: Time 225.00 seconds, asi/vm exits = 756191 / 2425297 = 31.18 %
KVM/VCPU 0xffffc90039f35000/3: Time 225.00 seconds, asi/vm exits = 521712 / 1051189 = 49.63 %
KVM/VCPU 0xffffc90039f35000/4: Time 224.91 seconds, asi/vm exits = 23353 / 73144 = 31.93 %
KVM/VCPU 0xffffc90039f35000/5: Time 224.93 seconds, asi/vm exits = 19609 / 60075 = 32.64 %
KVM/VCPU 0xffffc90039f35000/6: Time 224.93 seconds, asi/vm exits = 26320 / 81998 = 32.10 %
KVM/VCPU 0xffffc90039f35000/7: Time 224.99 seconds, asi/vm exits = 22509 / 85046 = 26.47 %
total_asi_exits = 2353420
```

# Initial Results - Redis

## Exit details

| RIP | data_addr | accessor | est_alloc_site | count | CDF |
|---|---|---|---|---|---|
| 0xffffffff811cecd3 | 0xffff88563e42c938 | el/sched/exclusive.c:7283 | PO: ./kernel/fork.c:1636 | 276673 | 1.000000 |
| 0xffffffff811cecd3 | 0xffff88554bc49938 | el/sched/exclusive.c:7283 | PO: ./kernel/events/core.c:10843 | 233775 | 0.887946 |
| 0xffffffff811c79b1 | 0xffffe8a0612b0070 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 151020 | 0.793267 |
| 0xffffffff811da155 | 0xffff885585e57c58 | el/sched/exclusive.c:7664 | ./net/core/skbuff.c:213 | 54685 | 0.732103 |
| 0xffffffff811c79b1 | 0xffffe8a0612f0070 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 45065 | 0.709956 |
| 0xffffffff81192686 | 0xffff88554bc49938 | ernel/sched/cputime.c:154 | PO: ./kernel/events/core.c:10843 | 37279 | 0.691704 |
| 0xffffffff811c79b1 | 0xffffe8a05ccf6cf0 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 32923 | 0.676606 |
| 0xffffffff81192686 | 0xffff88563e42c938 | ernel/sched/cputime.c:154 | PO: ./kernel/fork.c:1636 | 31714 | 0.663272 |
| 0xffffffff811da155 | 0xffff8855596c4c58 | el/sched/exclusive.c:7664 | ./net/core/skbuff.c:213 | 30228 | 0.650428 |
| 0xffffffff811ced4d | 0xffffffff83a2b930 | el/sched/exclusive.c:7315 | config_consume_rt_capacity | 29209 | 0.638185 |
| 0xffffffff811c79a2 | 0xffff885551c508d8 | rnel/sched/cpuacct.c:1284 | ./net/core/skbuff.c:213 | 24593 | 0.626356 |
| 0xffffffff815f0880 | 0xffff8854864b0380 | ./lib/llist.c:97 | ./fs/eventfd.c:658 | 24471 | 0.616395 |
| 0xffffffff811c79b1 | 0xffffe8a060a6dfe0 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 21122 | 0.606485 |
| 0xffffffff811c79b1 | 0xffffe8a060aece90 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 20673 | 0.597930 |

# Initial Results - Redis

## Exit details

| RIP | data_addr | accessor | est_alloc_site | count | CDF |
|---|---|---|---|---|---|
| 0xffffffff811cecd3 | 0xffff88563e42c938 | el/sched/exclusive.c:7283 | PO: ./kernel/fork.c:1636 | 276673 | 1.000000 |
| 0xffffffff811cecd3 | 0xffff88554bc49938 | el/sched/exclusive.c:7283 | PO: ./kernel/events/core.c:10843 | 233775 | 0.887946 |
| 0xffffffff811c79b1 | 0xffffe8a0612b0070 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 151020 | 0.793267 |
| 0xffffffff811da155 | 0xffff885585e57c58 | el/sched/exclusive.c:7664 | ./net/core/skbuff.c:213 | 54685 | 0.732103 |
| 0xffffffff811c79b1 | 0xffffe8a0612f0070 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 45065 | 0.709956 |
| 0xffffffff81192686 | 0xffff88554bc49938 | ernel/sched/cputime.c:154 | PO: ./kernel/events/core.c:10843 | 37279 | 0.691704 |
| 0xffffffff811c79b1 | 0xffffe8a05ccf6cf0 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 32923 | 0.676606 |
| 0xffffffff81192686 | 0xffff88563e42c938 | ernel/sched/cputime.c:154 | PO: ./kernel/fork.c:1636 | 31714 | 0.663272 |
| 0xffffffff811da155 | 0xffff8855596c4c58 | el/sched/exclusive.c:7664 | ./net/core/skbuff.c:213 | 30228 | 0.650428 |
| 0xffffffff811ced4d | 0xffffffff83a2b930 | el/sched/exclusive.c:7315 | config_consume_rt_capacity | 29209 | 0.638185 |
| 0xffffffff811c79a2 | 0xffff885551c508d8 | rnel/sched/cpuacct.c:1284 | ./net/core/skbuff.c:213 | 24593 | 0.626356 |
| 0xffffffff815f0880 | 0xffff8854864b0380 | ./lib/llist.c:97 | ./fs/eventfd.c:658 | 24471 | 0.616395 |
| 0xffffffff811c79b1 | 0xffffe8a060a6dfe0 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 21122 | 0.606485 |
| 0xffffffff811c79b1 | 0xffffe8a060aece90 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 20673 | 0.597930 |

```
7278        curr->se.exec_start = now;
7279        schedstat_set(curr->se.statistics.exec_max,
7280                      max(curr->se.statistics.exec_max, delta_exec));
7281
7282        curr->se.sum_exec_runtime += delta_exec;
7283        account_group_exec_runtime(curr, delta_exec);
```

# Initial Results - Redis

## Exit details

| RIP | data_addr | accessor | est_alloc_site | count | CDF |
|---|---|---|---|---|---|
| 0xffffffff811cecd3 | 0xffff88563e42c938 | el/sched/exclusive.c:7283 | PO: ./kernel/fork.c:1636 | 276673 | 1.000000 |
| 0xffffffff811cecd3 | 0xffff88554bc49938 | el/sched/exclusive.c:7283 | PO: ./kernel/events/core.c:10843 | 233775 | 0.887946 |
| 0xffffffff811c79b1 | 0xffffe8a0612b0070 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 151020 | 0.793267 |
| 0xffffffff811da155 | 0xffff885585e57c58 | el/sched/exclusive.c:7664 | ./net/core/skbuff.c:213 | 54685 | 0.732103 |
| 0xffffffff811c79b1 | 0xffffe8a0612f0070 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 45065 | 0.709956 |
| 0xffffffff81192686 | 0xffff88554bc49938 | ernel/sched/cputime.c:154 | PO: ./kernel/events/core.c:10843 | 37279 | 0.691704 |
| 0xffffffff811c79b1 | 0xffffe8a05ccf6cf0 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 32923 | 0.676606 |
| 0xffffffff81192686 | 0xffff88563e42c938 | ernel/sched/cputime.c:154 | PO: ./kernel/fork.c:1636 | 31714 | 0.663272 |
| 0xffffffff811da155 | 0xffff8855596c4c58 | el/sched/exclusive.c:7664 | ./net/core/skbuff.c:213 | 30228 | 0.650428 |
| 0xffffffff811ced4d | 0xffffffff83a2b930 | el/sched/exclusive.c:7315 | config_consume_rt_capacity | 29209 | 0.638185 |
| 0xffffffff811c79a2 | 0xffff885551c508d8 | rnel/sched/cpuacct.c:1284 | ./net/core/skbuff.c:213 | 24593 | 0.626356 |
| 0xffffffff815f0880 | 0xffff8854864b0380 | ./lib/llist.c:97 | ./fs/eventfd.c:658 | 24471 | 0.616395 |
| 0xf | | | | 21122 | 0.606485 |
| 0xf | | | | 20673 | 0.597930 |

```
1628 static int copy_signal(unsigned long clone_flags, struct task_struct *tsk)
1629 {
1630         struct signal_struct *sig;
1631
1632         if (clone_flags & CLONE_THREAD)
1633                 return 0;
1634
1635 #ifdef CONFIG_ADDRESS_SPACE_ISOLATION
1636         sig = kzalloc(sizeof(struct signal_struct),
1637                         GFP_KERNEL | GFP_NONSENSITIVE);
 NORMAL  PASTE  kernel/fork.c                                    55%
```

# Initial Results - Redis

## Exit details by allocation site

| variable | count | CDF |
|---|---|---|
| PO: ./mm/percpu-vm.c:284 | 760078 | 1.000000 |
| PO: ./kernel/fork.c:1636 | 319451 | 0.692166 |
| PO: ./kernel/events/core.c:10843 | 293764 | 0.562787 |
| ./net/core/skbuff.c:213 | 208683 | 0.443812 |
| PO: ./kernel/fork.c:249 | 193298 | 0.359294 |
| PO: ./kernel/sched/topology.c:1766 | 157080 | 0.281008 |
| ./kernel/fork.c:1860 | 63355 | 0.217390 |

| RIP | data_addr | accessor | est_alloc_site | count |
|---|---|---|---|---|
| 0xffffffff811c79b1 | 0xffffe8a0612b0070 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 151020 |
| 0xffffffff811c79b1 | 0xffffe8a0612f0070 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 45065 |
| 0xffffffff811c79b1 | 0xffffe8a05ccf6cf0 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 32923 |
| 0xffffffff811c79b1 | 0xffffe8a060a6dfe0 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 21122 |
| 0xffffffff811c79b1 | 0xffffe8a060aece90 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 20673 |
| 0xffffffff811c79b1 | 0xffffe8a05ccb6cf0 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 20118 |
| 0xffffffff811c79b1 | 0xffffe8a05cc36cf0 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 19819 |
| 0xffffffff811c79b1 | 0xffffe8a060ab0070 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 14848 |
| 0xffffffff8120541c | 0xffffe8a05b682f40 | kernel/rcu/srcutree.c:418 | PO: ./mm/percpu-vm.c:284 | 14166 |
| 0xffffffff811c79b1 | 0xffffe8a05cc76cf0 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 13879 |
| 0xffffffff811c79b1 | 0xffffe8a0612adfe0 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 13765 |
| 0xffffffff811c79b1 | 0xffffe8a060a2dfe0 | rnel/sched/cpuacct.c:1284 | PO: ./mm/percpu-vm.c:284 | 12276 |

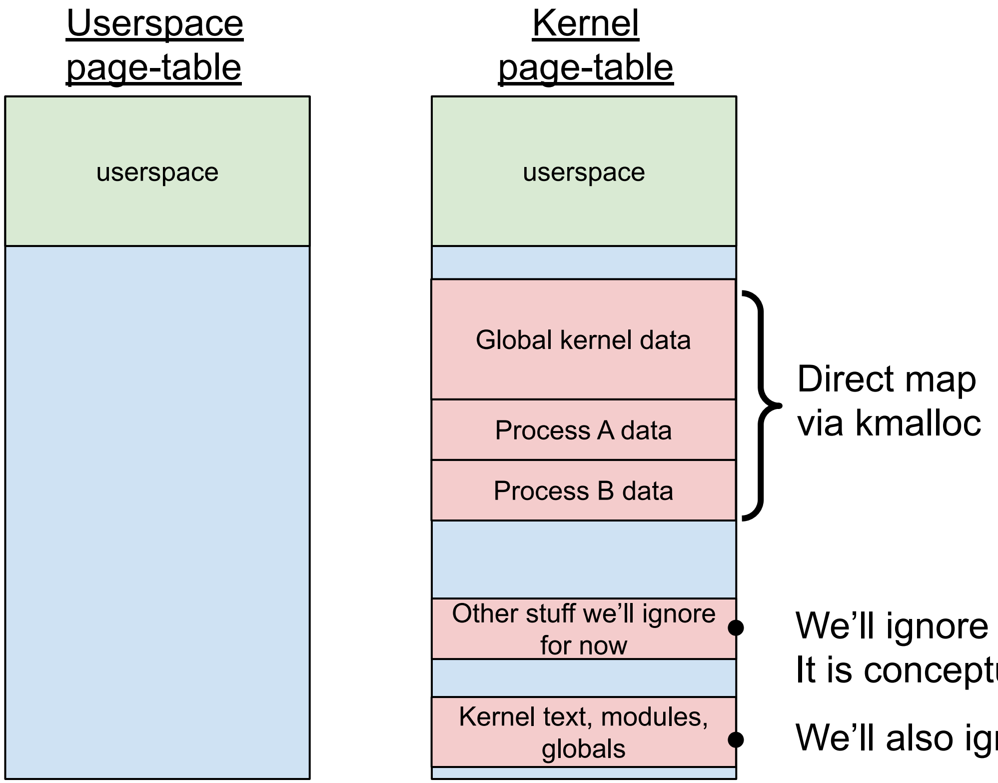
# Challenges in managing dynamic memory

1. How to manage different allocations
    a. kmalloc
    b. vmalloc
    c. per-cpu
2. What does it mean for data to be non-sensitive?
    a. Is memory non-sensitive for the current VM or system wide?

# The KPTI Model - Control & Data Privilege

Not mapped

Privileged data



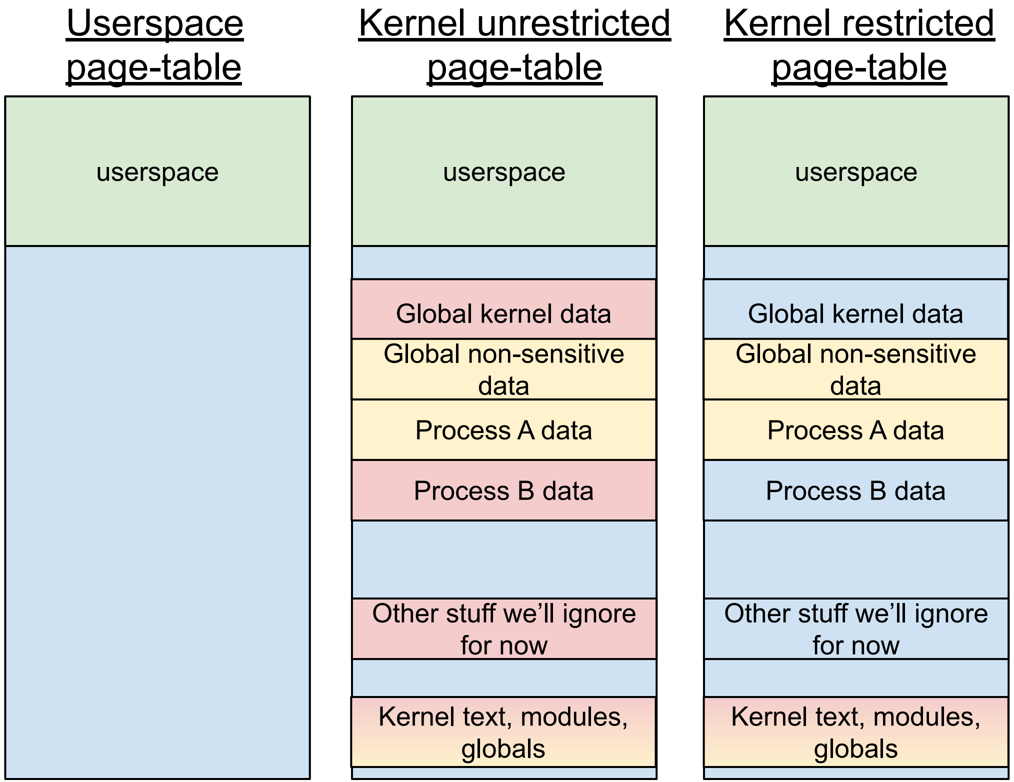

To mitigate Meltdown attacks, KPTI differentiates between privileged/unprivileged execution level.

The methodology - using two page tables to separate between user space memory and kernel privileged memory.

We'll ignore vmalloc space for now.
It is conceptually similar to direct map

We'll also ignore global vars

# The ASI Model - Data Privilege


In ASI, we define privilege based on data access, not execution-level. We add another "restricted" page-table which only maps kernel **non-sensitive** data.

Data is deemed **non-sensitive** if, when stolen by a malicious VM, does not pose a security threat to other VMs or cloud's infrastructure.

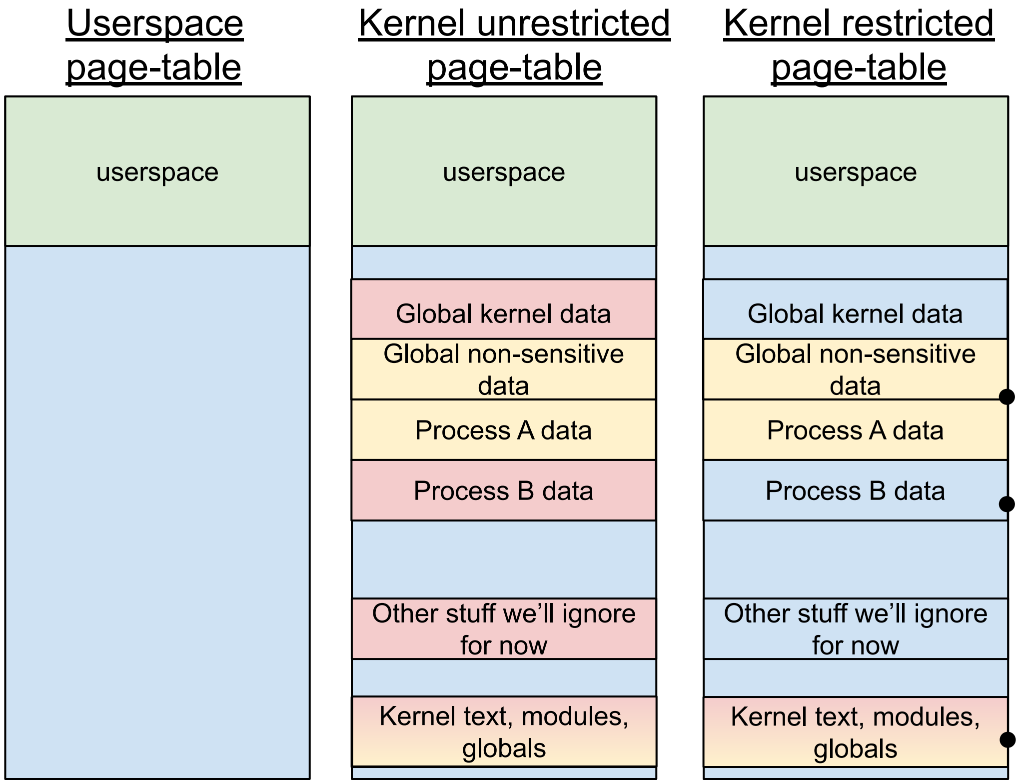
For performance reasons, we're interested in memory that is accessed frequently by the kernel, when operating a VM between VMEXIT and VMENTER.

# The ASI Model - Data Privilege

Not mapped

Privileged data

Non-sensitive data




Non-sensitive data can be accessed freely, without the need for any L1TF mitigations

Access to "unmapped" area will cause a PF, which will switch to the unrestricted page-table. Use L1TF mitigation when switching (stunning/L1D-flush)

Ignore for now

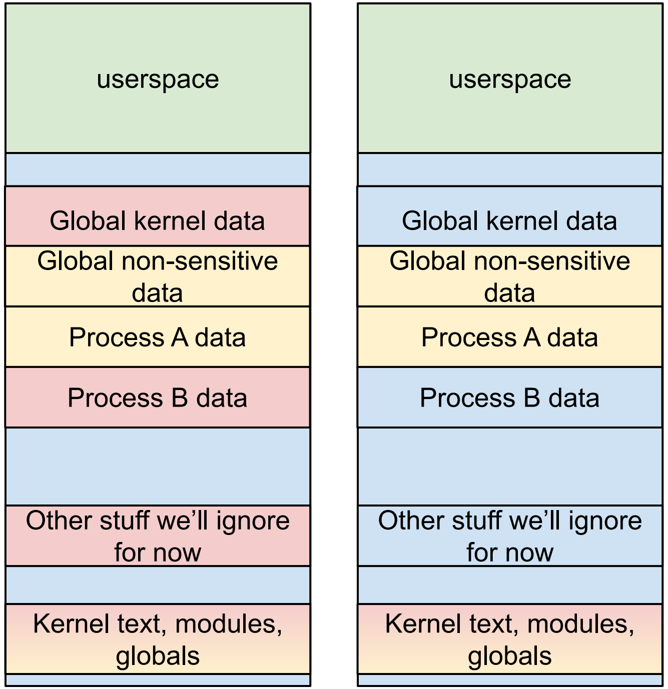
# Data Privilege - The Locality Dilemma

Not mapped

Privileged data

Non-sensitive data

Kernel unrestricted page-table | Kernel restricted page-table




**Challenge 1**

Is data considered non-sensitive locally in a process or globally in the entire system?

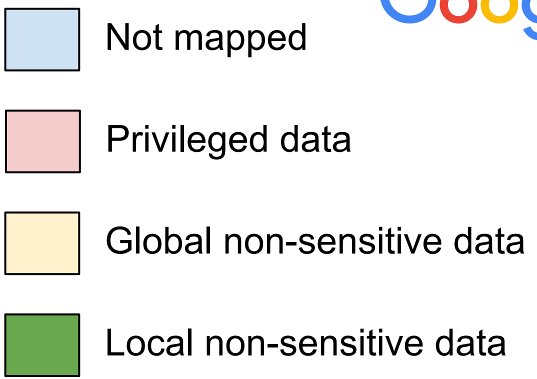
Examples:

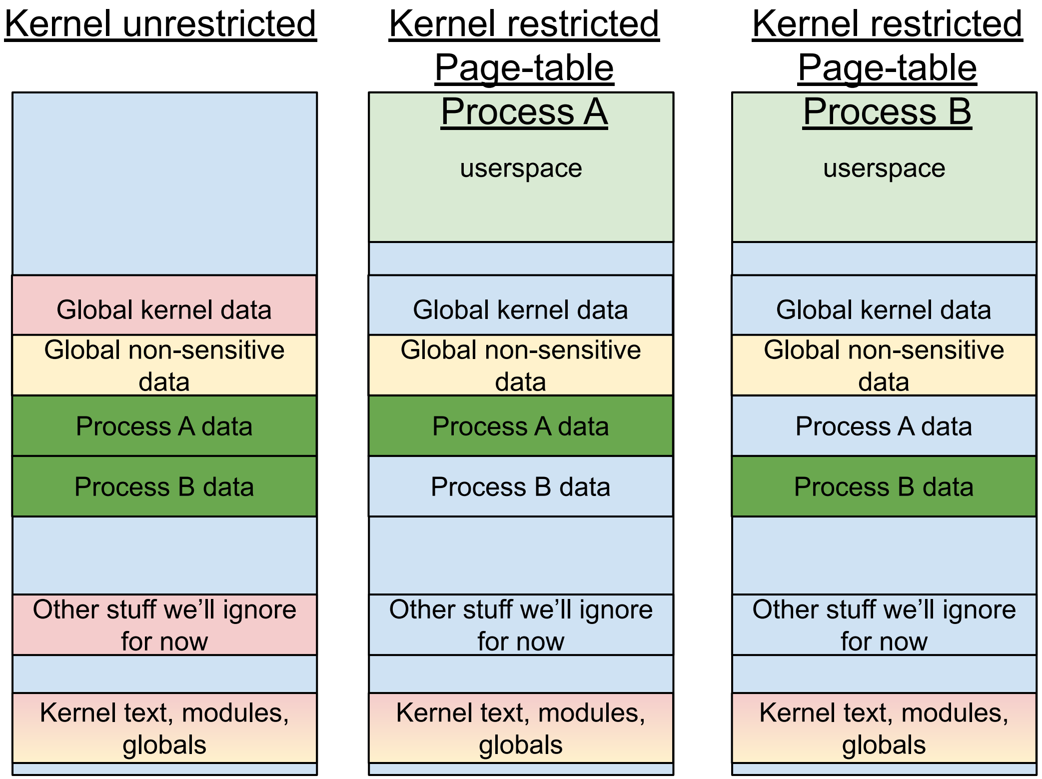
1. Local data: VMCS, vcpu, file-descriptor-table
2. Global data: sk_buffs

All non-sensitive data in ASI can be read by a guest VM via an L1TF attack

While we want VM-1 to access its VMCS freely we don't want VM-1 to read the VMCS of VM-2!!

# Partitioning Global/Local Data

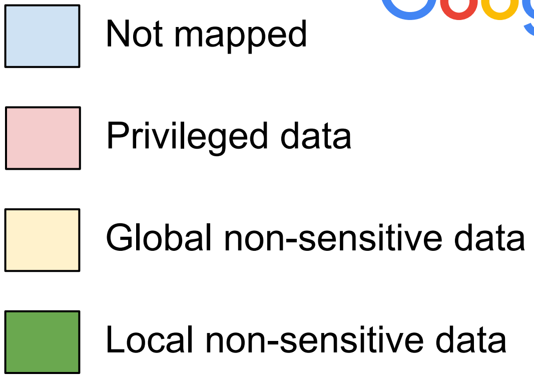


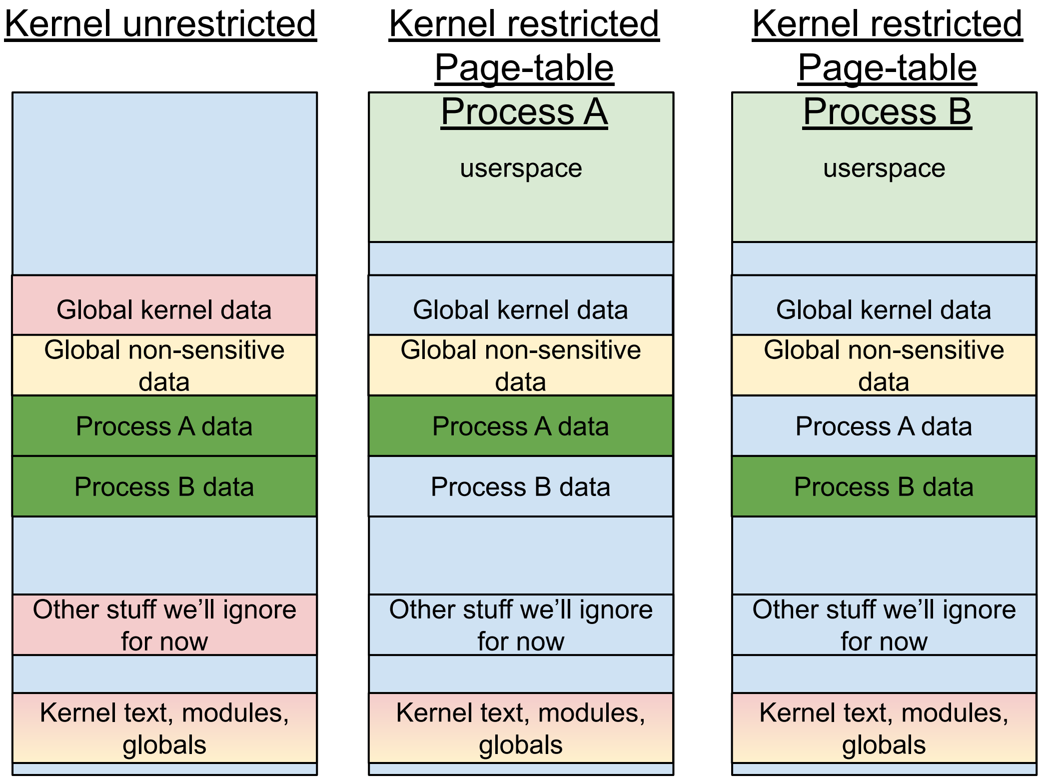




**Solution 1**

Map local non-sensitive data to the specific process restricted table. Map global non-sensitive data to any ASI restricted table.

# Partitioning Global/Local Data



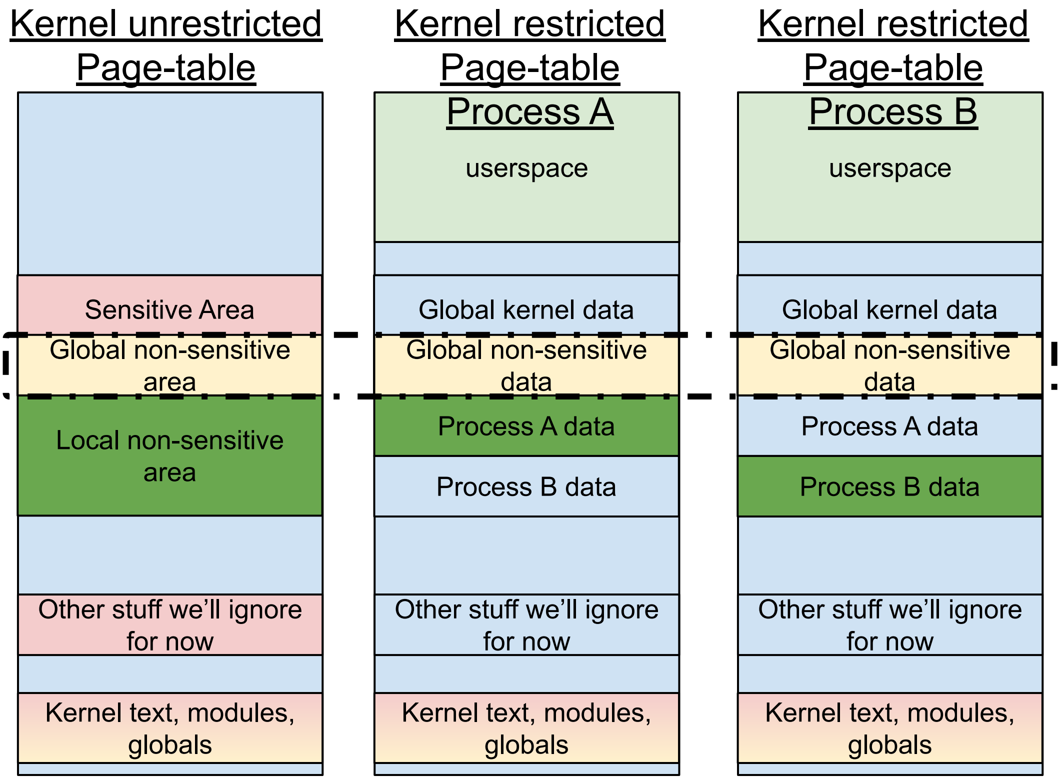



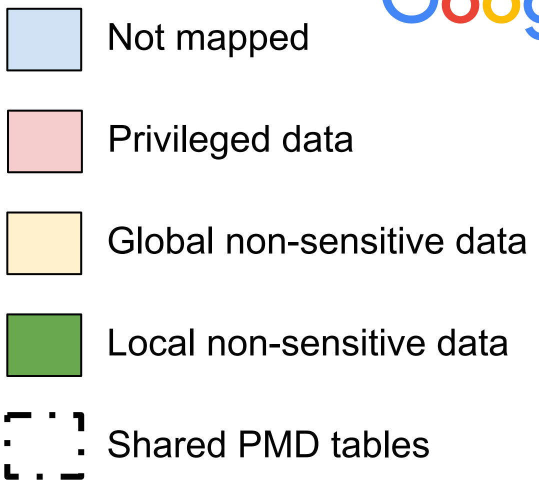

**Challenge 2:**
What happens when we allocate global non-sensitive data?

We need to update the page-tables of ALL processes on every allocation :(

That can be prohibitively slow, depending on how many processes are running ASI.

# Sharing Global Data Entries




**Solution 2**

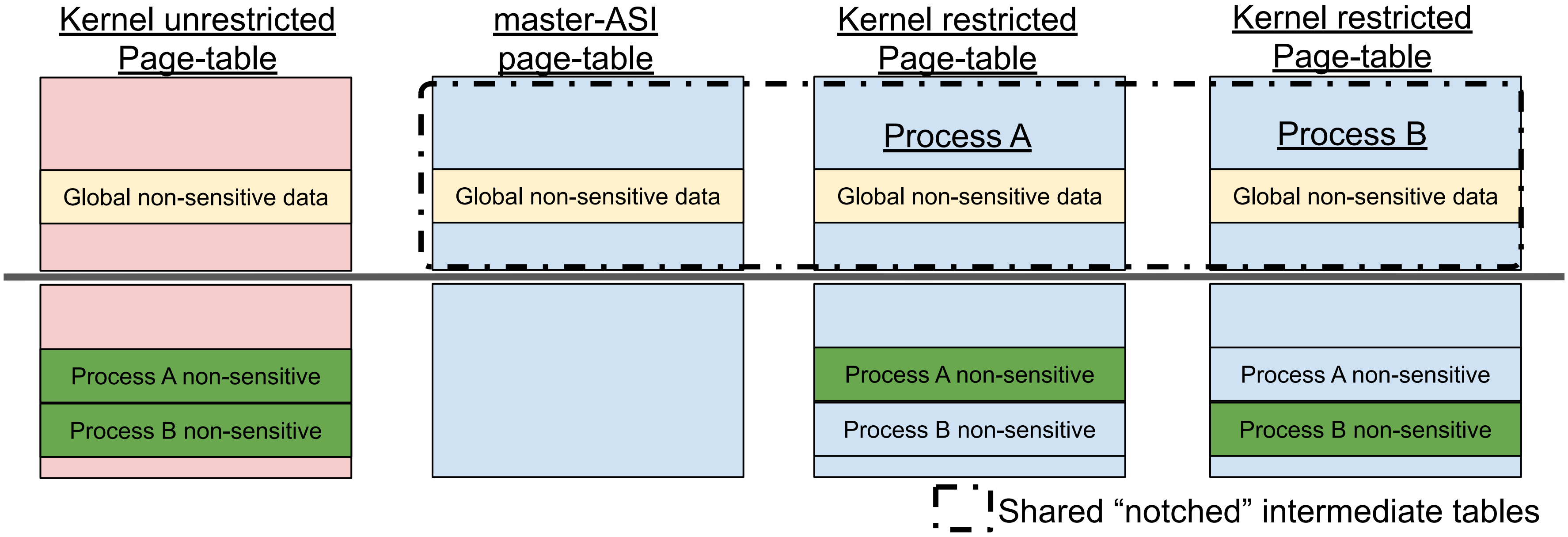
Divide dynamic memory area into 2 regions:

1. Global non-sensitive
2. Local non-sensitive

To avoid constant update of global non-sensitive area in all processes - share the PUD entries

# Dynamic Memory - Synchronization

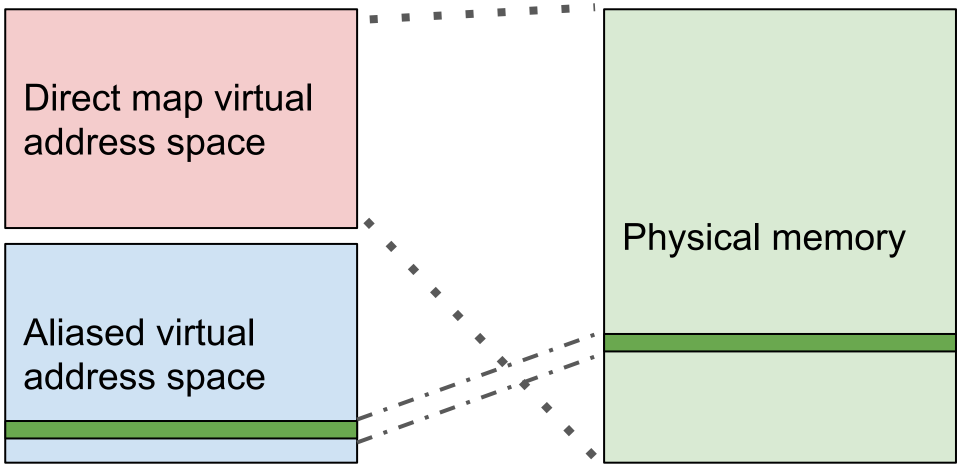
We manage all global non-sensitive allocations in a single “master-ASI” table.
If we get a PF in the global area, we pull the shared higher level page-table entry into the process ASI restricted-table



# Dynamic Memory - Aliasing

Alternative approach to partitioning the direct map

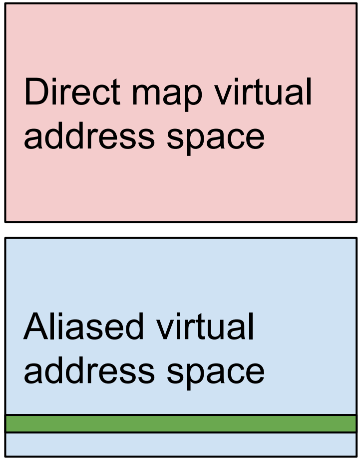
Partition the kernel address range dedicated to the direct map into two equal halves, with the upper half being an alias of the regular direct map.



# Dynamic Memory - Aliasing

Alternative approach to partitioning the direct map

Partition the kernel address range dedicated to the direct map into two equal halves, with the upper half being an alias of the regular direct map.



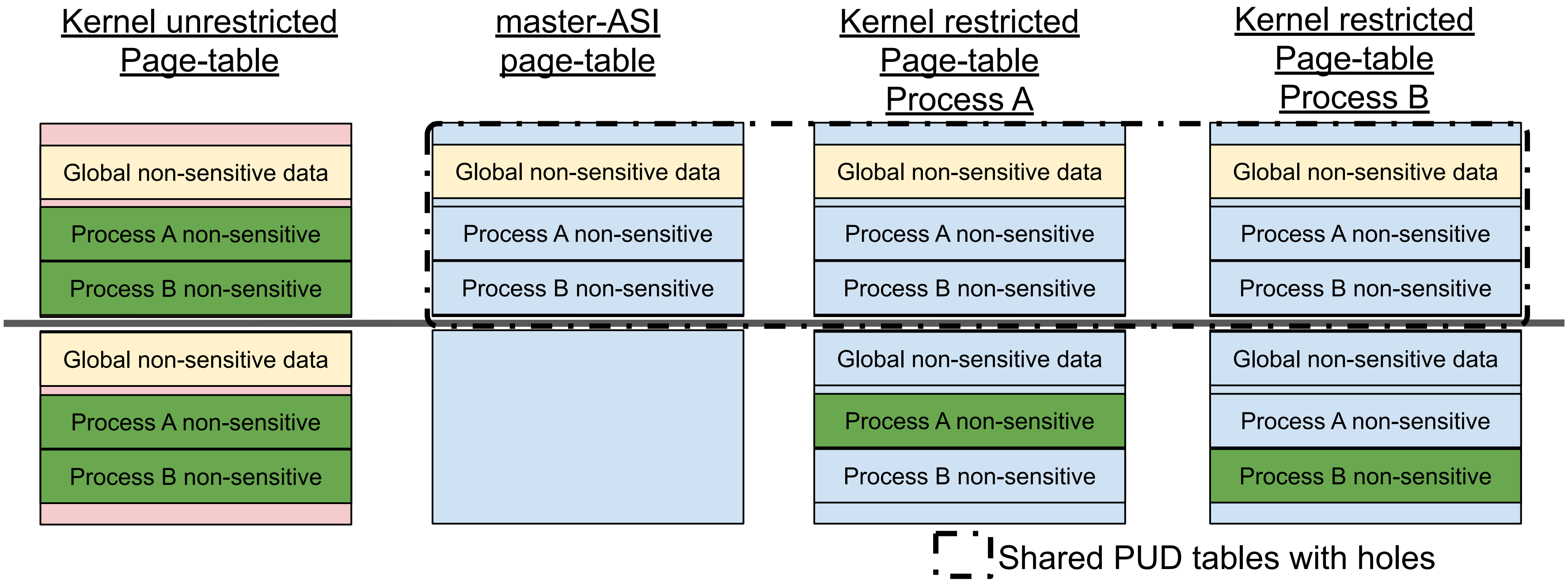

kmalloc(size, ASI_FLAG) → address in aliased space

phys_to_virt/virt_to_phys etc. modified

Reduces max supported RAM size by half, if implemented in a straightforward way

# Dynamic Memory - Aliasing

In the restricted page tables, the aliased direct map only has local non-sensitive mappings, while the regular direct map only has global non-sensitive mappings

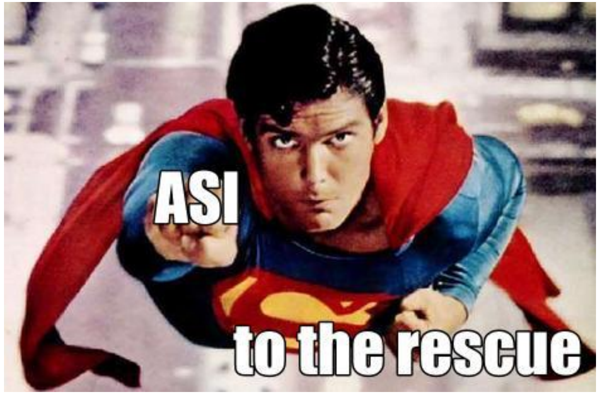


# Connecting All Efforts Together

- Eventually, Linux needs one complete solution
- Many similar use-cases - we should strive to merge
  - We need “union” of functionality for full protection
- There should be one paradigm/infrastructure to deal with
  - Per-process memory
  - Namespace memory
  - VM memory
  - Global non-sensitive memory

# Summary - efficiently defeating speculative attacks

1. ASI redefines access-control based on the data
    a. Namely, sensitive vs. non-sensitive **data**
    b. Instead of based on control-flow: **userspace vs. kernel**
2. A allow-list approach is more sustainable than block-list
3. Apply expensive (e.g., L1D flush, stunning) mitigations only when necessary
    a. Yields a complete **and** efficient solution
4. Can extend KPTI model and even improve performance
5. We want to integrate with concurrent efforts!



# aerospike_ycsb

```
KVM/VCPU 0xffffc9006ec95000/0: Time 155.94 seconds, asi/vm exits = 127380 / 5336308 = 2.39 %
KVM/VCPU 0xffffc9006ec95000/1: Time 155.75 seconds, asi/vm exits = 158369 / 513023 = 30.87 %
KVM/VCPU 0xffffc9006ec95000/2: Time 155.75 seconds, asi/vm exits = 121171 / 364944 = 33.20 %
KVM/VCPU 0xffffc9006ec95000/3: Time 155.75 seconds, asi/vm exits = 106071 / 401861 = 26.39 %
KVM/VCPU 0xffffc9006ec95000/4: Time 155.75 seconds, asi/vm exits = 122137 / 333260 = 36.65 %
KVM/VCPU 0xffffc9006ec95000/5: Time 155.75 seconds, asi/vm exits = 108123 / 325609 = 33.21 %
KVM/VCPU 0xffffc9006ec95000/6: Time 155.75 seconds, asi/vm exits = 102944 / 283367 = 36.33 %
KVM/VCPU 0xffffc9006ec95000/7: Time 155.75 seconds, asi/vm exits = 97132 / 272972 = 35.58 %
KVM/VCPU 0xffffc900338bd000/0: Time 246.45 seconds, asi/vm exits = 115375 / 5344884 = 2.16 %
KVM/VCPU 0xffffc900338bd000/1: Time 227.32 seconds, asi/vm exits = 170465 / 566336 = 30.10 %
KVM/VCPU 0xffffc900338bd000/2: Time 227.35 seconds, asi/vm exits = 106785 / 306784 = 34.81 %
KVM/VCPU 0xffffc900338bd000/3: Time 227.28 seconds, asi/vm exits = 118105 / 397094 = 29.74 %
KVM/VCPU 0xffffc900338bd000/4: Time 227.34 seconds, asi/vm exits = 122201 / 336527 = 36.31 %
KVM/VCPU 0xffffc900338bd000/5: Time 227.27 seconds, asi/vm exits = 116264 / 454567 = 25.58 %
KVM/VCPU 0xffffc900338bd000/6: Time 227.33 seconds, asi/vm exits = 117845 / 315211 = 37.39 %
KVM/VCPU 0xffffc900338bd000/7: Time 227.26 seconds, asi/vm exits = 120306 / 329583 = 36.50 %
```

| ip | address | allocator | count | function | variable | mem_type | allocation | CDF |
|---|---|---|---|---|---|---|---|---|
| 0xffffffff811cecd3 | 0xffff885499d07bf8 | 0xffffffff8114641f | 72811 | el/sched/exclusive.c:7282 | PO: ./kernel/fork.c:1623 | direct mapping | ./kernel/fork.c:1623 | 1.000000 |
| 0xffffffff811c7959 | 0xffff8855cccbc138 | 0xffffffff8131373e | 57472 | rnel/sched/cpuacct.c:1266 | ./kernel/cgroup/cgroup.c:5116 | direct mapping | ./kernel/events/core.c:8138 | 0.946175 |
| 0xffffffff811cecd3 | 0xffff88554f231bf8 | 0xffffffff8114641f | 54673 | el/sched/exclusive.c:7282 | PO: ./kernel/fork.c:1623 | direct mapping | ./kernel/fork.c:1623 | 0.903689 |
| 0xffffffff811db7c5 | 0xffffffff83a2b930 | 0x9 | 38278 | el/sched/exclusive.c:7220 | config_discount_busy_poll_time | kernel text map | <Error: 0x9> | 0.863272 |
| 0xffffffff81192696 | 0xffff88554f231bf8 | 0xffffffff8114641f | 29422 | ernel/sched/cputime.c:154 | PO: ./kernel/fork.c:1623 | direct mapping | ./kernel/fork.c:1623 | 0.834976 |

# aerospike_ycsb

```
KVM/VCPU 0xffffc9006ec95000/0: Time 155.94 seconds, asi/vm exits = 127380 / 5336308 = 2.39 %
KVM/VCPU 0xffffc9006ec95000/1: Time 155.75 seconds, asi/vm exits = 158369 / 513023 = 30.87 %
KVM/VCPU 0xffffc9006ec95000/2: Time 155.75 seconds, asi/vm exits = 121171 / 364944 = 33.20 %
KVM/VCPU 0xffffc9006ec95000/3: Time 155.75 seconds, asi/vm exits = 106071 / 401861 = 26.39 %
KVM/VCPU 0xffffc9006ec95000/4: Time 155.75 seconds, asi/vm exits = 122137 / 333260 = 36.65 %
KVM/VCPU 0xffffc9006ec95000/5: Time 155.75 seconds, asi/vm exits = 108123 / 325609 = 33.21 %
KVM/VCPU 0xffffc9006ec95000/6: Time 155.75 seconds, asi/vm exits = 102944 / 283367 = 36.33 %
KVM/VCPU 0xffffc9006ec95000/7: Time 155.75 seconds, asi/vm exits = 97132 / 272972 = 35.58 %
KVM/VCPU 0xffffc ... 344884 = 2.16 %
KVM/VCPU 0xffffc ... 66336 = 30.10 %
KVM/VCPU 0xffffc ... 06784 = 34.81 %
KVM/VCPU 0xffffc ... 97094 = 29.74 %
KVM/VCPU 0xffffc ... 36527 = 36.31 %
KVM/VCPU 0xffffc ... 54567 = 25.58 %
KVM/VCPU 0xffffc ... 15211 = 37.39 %
KVM/VCPU 0xffffc ... 29583 = 36.50 %
```

| variable | count | CDF |
|---|---|---|
| cluster_masks | 349794 | 1.000000 |
| PO: ./kernel/fork.c:1623 | 181553 | 0.741417 |
| PO: ./kernel/fork.c:241 | 128836 | 0.607205 |
| ./net/core/skbuff.c:213 | 93792 | 0.511964 |
| ipi_mask | 82907 | 0.442628 |

| ip | address | allocator | count | function | variable | mem_type | allocation | CDF |
|---|---|---|---|---|---|---|---|---|
| 0xffffffff811cecd3 | 0xffff885499d07bf8 | 0xffffffff8114641f | 72811 | el/sched/exclusive.c:7282 | PO: ./kernel/fork.c:1623 | direct mapping | ./kernel/fork.c:1623 | 1.000000 |
| 0xffffffff811c7959 | 0xffff8855cccbc138 | 0xffffffff8131373e | 57472 | rnel/sched/cpuacct.c:1266 | ./kernel/cgroup/cgroup.c:5116 | direct mapping | ./kernel/events/core.c:8138 | 0.946175 |
| 0xffffffff811cecd3 | 0xffff88554f231bf8 | 0xffffffff8114641f | 54673 | el/sched/exclusive.c:7282 | PO: ./kernel/fork.c:1623 | direct mapping | ./kernel/fork.c:1623 | 0.903689 |
| 0xffffffff811db7c5 | 0xffffffff83a2b930 | 0x9 | 38278 | el/sched/exclusive.c:7220 | config_discount_busy_poll_time | kernel text map | <Error: 0x9> | 0.863272 |
| 0xffffffff81192696 | 0xffff88554f231bf8 | 0xffffffff8114641f | 29422 | ernel/sched/cputime.c:154 | PO: ./kernel/fork.c:1623 | direct mapping | ./kernel/fork.c:1623 | 0.834976 |

# netperf

```
KVM/VCPU 0xffffc90035b71000/0: Time 425.39 seconds, asi/vm exits = 214111 / 5259680 = 4.07 %
KVM/VCPU 0xffffc90035b71000/1: Time 425.02 seconds, asi/vm exits = 247286 / 542627 = 45.57 %
KVM/VCPU 0xffffc90035b71000/2: Time 425.27 seconds, asi/vm exits = 245862 / 901932 = 27.26 %
KVM/VCPU 0xffffc90035b71000/3: Time 425.27 seconds, asi/vm exits = 288560 / 6982195 = 4.13 %
KVM/VCPU 0xffffc90035b71000/4: Time 425.26 seconds, asi/vm exits = 281123 / 5741351 = 4.90 %
KVM/VCPU 0xffffc90035b71000/5: Time 425.07 seconds, asi/vm exits = 206582 / 332710 = 62.09 %
KVM/VCPU 0xffffc90035b71000/6: Time 425.23 seconds, asi/vm exits = 207339 / 324566 = 63.88 %
KVM/VCPU 0xffffc90035b71000/7: Time 425.15 seconds, asi/vm exits = 337102 / 5772802 = 5.84 %
total_asi_exits = 2027965
KVM/VCPU 0xffffc90036131000/0: Time 518.22 seconds, asi/vm exits = 238809 / 10276123 = 2.32 %
KVM/VCPU 0xffffc90036131000/1: Time 518.82 seconds, asi/vm exits = 350573 / 2138048 = 16.40 %
KVM/VCPU 0xffffc90036131000/2: Time 518.80 seconds, asi/vm exits = 220670 / 385801 = 57.20 %
KVM/VCPU 0xffffc90036131000/3: Time 518.77 seconds, asi/vm exits = 243547 / 2612429 = 9.32 %
KVM/VCPU 0xffffc90036131000/4: Time 518.12 seconds, asi/vm exits = 330598 / 1246831 = 26.52 %
KVM/VCPU 0xffffc90036131000/5: Time 518.83 seconds, asi/vm exits = 239431 / 3858650 = 6.21 %
KVM/VCPU 0xffffc90036131000/6: Time 518.80 seconds, asi/vm exits = 210698 / 344824 = 61.10 %
KVM/VCPU 0xffffc90036131000/7: Time 518.80 seconds, asi/vm exits = 231221 / 2521580 = 9.17 %
total_asi_exits = 2065547
```

| ip | address | allocator | count | function | variable | mem_type | allocation | CDF |
|---|---|---|---|---|---|---|---|---|
| 0xffffffff811c79a2 | 0xffff88547a3428d8 | 0xffffffff81307868 | 82179 | rnel/sched/cpuacct.c:1282 | ./net/core/skbuff.c:213 | direct mapping | ./kernel/events/core.c:10843 | 1.000000 |
| 0xffffffff811cecd3 | 0xffff885537e77bf8 | 0xffffffff8114641f | 60940 | el/sched/exclusive.c:7282 | PO: ./kernel/fork.c:1623 | direct mapping | ./kernel/fork.c:1623 | 0.932457 |
| 0xffffffff811da155 | 0xffff885627b4fc58 | 0xffffffff8196e5fd | 32622 | el/sched/exclusive.c:7663 | ./net/core/skbuff.c:213 | direct mapping | ./net/core/skbuff.c:213 | 0.882370 |
| | 0xffff8855b3980c58 | 0xffffffff8196e5fd | 28198 | el/sched/exclusive.c:7663 | ./net/core/skbuff.c:213 | direct mapping | ./net/core/skbuff.c:213 | 0.855558 |
| | 0xffff8856d8018c58 | 0xffffffff81496be4 | 26941 | el/sched/exclusive.c:7663 | ./net/core/skbuff.c:213 | direct mapping | ./fs/proc/proc_sysctl.c:1323 | 0.832382 |

# netperf

```
KVM/VCPU 0xffffc90035b71000/0: Time 425.39 seconds, asi/vm exits = 214111 / 5259680 = 4.07 %
KVM/VCPU 0xffffc90035b71000/1: Time 425.02 seconds, asi/vm exits = 247286 / 542627 = 45.57 %
KVM/VCPU 0xffffc90035b71000/2: Time 425.27 seconds, asi/vm exits = 245862 / 901932 = 27.26 %
KVM/VCPU 0xffffc90035b71000/3: Time 425.27 seconds, asi/vm exits = 288560 / 6982195 = 4.13 %
KVM/VCPU 0xffffc90035b71000/4: Time 425.26 seconds, asi/vm exits = 281123 / 5741351 = 4.90 %
KVM/VCPU 0xffffc90035b71000/5: Time 425.07 seconds, asi/vm exits = 206582 / 332710 = 62.09 %
KVM/VCPU 0xffffc90035b71000/6: Time 425.23 seconds, asi/vm exits = 207339 / 324566 = 63.88 %
KVM/VCPU 0xffffc90035b71000/7: Time 425.15 seconds, asi/vm exits = 337102 / 5772802 = 5.84 %
total_asi_exits = 2027965
KVM/VCPU 0xffffc90036131000/0: Time 518.22 seconds, asi/vm exits = 238809 / 10276123 = 2.32 %
KVM/VCPU 0xffffc90036131000/1: Time 518.82 seconds, asi/vm exits = 350573 / 2138048 = 16.40 %
KVM/VCPU 0xffffc90036131000/2: Time 518.80 seconds, asi/vm exits = 220670 / 385801 = 57.20 %
KVM/VCPU 0xffffc90036131000/3: Time 518.77 seconds, asi/vm exits = 243547 / 2612429 = 9.32 %
KVM/VCPU 0xf                                                          246831 = 26.52 %
KVM/VCPU 0xf                                                          858650 = 6.21 %
KVM/VCPU 0xf                                                          44824 = 61.10 %
KVM/VCPU 0xf                                                          521580 = 9.17 %
total_asi_ex
```

```
                                          count      CDF
variable
./net/core/skbuff.c:213                  310674 1.000000
PO: ./mm/percpu-vm.c:284                 182433 0.744656
PO: ./kernel/fork.c:1623                 122907 0.594714
```

```
                                                         count                    function                   variable        mem_type                     allocation      CDF
ip                 address            allocator
0xffffffff811c79a2 0xffff88547a3428d8 0xffffffff81307868 82179 rnel/sched/cpuacct.c:1282     ./net/core/skbuff.c:213 direct mapping     ./kernel/events/core.c:10843 1.000000
0xffffffff811cecd3 0xffff885537e77bf8 0xffffffff8114641f 60940 el/sched/exclusive.c:7282   PO: ./kernel/fork.c:1623 direct mapping            ./kernel/fork.c:1623 0.932457
0xffffffff811da155 0xffff885627b4fc58 0xffffffff8196e5fd 32622 el/sched/exclusive.c:7663     ./net/core/skbuff.c:213 direct mapping         ./net/core/skbuff.c:213 0.882370
                   0xffff8855b3980c58 0xffffffff8196e5fd 28198 el/sched/exclusive.c:7663     ./net/core/skbuff.c:213 direct mapping         ./net/core/skbuff.c:213 0.855558
                   0xffff8856d8018c58 0xffffffff81496be4 26941 el/sched/exclusive.c:7663     ./net/core/skbuff.c:213 direct mapping   ./fs/proc/proc_sysctl.c:1323 0.832382
```

# Which Accesses Cause ASI-exits

| ip | address | function | variable | count | CDF |
|---|---|---|---|---|---|
| 0xffffffff813bdbf1 | 0xffff88556abc1398 | ./mm/memcontrol.c:886 | <Unknown alloc> | 140972 | 1.000000 |
| 0xffffffff81194826 | 0xffff88556f858bf8 | ernel/sched/cputime.c:148 | PO: ./kernel/fork.c:1620 | 51283 | 0.641108 |
| 0xffffffff811de7c3 | 0xffff885554673a20 | el/locking/lockdep.c:3311 | <Unknown alloc> | 28769 | 0.510550 |
| 0xffffffff81029b0d | 0xffff885556332bf0 | /virt/kvm/kvm_main.c:5152 | ./arch/x86/kvm/../../../virt/kvm/eventfd.c:1016 | 10452 | 0.437309 |
| 0xffffffff811ae0d1 | 0xffff8856e617ce84 | kernel/sched/fair.c:15198 | PO: ./kernel/sched/topology.c:1662 | 5422 | 0.410700 |
| ... | ... | ... | ... | ... | ... |
| 0xffffffff81366382 | 0xffff88559eb65f60 | ./mm/gup.c:2174 | ./arch/x86/mm/pgtable.c:30 | 1 | 0.000013 |
| 0xffffffff81366382 | 0xffff88559eb65f80 | ./mm/gup.c:2174 | ./arch/x86/mm/pgtable.c:30 | 1 | 0.000010 |
| 0xffffffff81366382 | 0xffff88559eb88620 | ./mm/gup.c:2174 | ./arch/x86/mm/pgtable.c:30 | 1 | 0.000008 |
| 0xffffffff81366382 | 0xffff88559eb88800 | ./mm/gup.c:2174 | ./arch/x86/mm/pgtable.c:30 | 1 | 0.000005 |
| 0xffffffff81366382 | 0xffff88559eb32110 | ./mm/gup.c:2174 | ./arch/x86/mm/pgtable.c:30 | 1 | 0.000003 |

```
876 struct mem_cgroup *mem_cgroup_from_task(struct task_struct *p)
877 {
878         /*
879          * mm_update_next_owner() may clear mm->owner to NULL
880          * if it races with swapoff, page migration, etc.
881          * So this can be called with p == NULL.
882          */
883         if (unlikely(!p))
884                 return NULL;
885 
886         return mem_cgroup_from_css(task_css(p, memory_cgrp_id));
887 }
NORMAL  mm/memcontrol.c
```

# Which Accesses Cause ASI-exits

| ip | address | function | variable | count | CDF |
| --- | --- | --- | --- | --- | --- |
| 0xffffffff813bdbf1 | 0xffff88556abc1398 | ./mm/memcontrol.c:886 | <Unknown alloc> | 140972 | 1.000000 |
| 0xffffffff81194826 | 0xffff88556f858bf8 | ernel/sched/cputime.c:148 | PO: ./kernel/fork.c:1620 | 51283 | 0.641108 |
| 0xffffffff811de7c3 | 0xffff885554673a20 | el/locking/lockdep.c:3311 | <Unknown alloc> | 28769 | 0.510550 |
| 0xffffffff81029b0d | 0xffff885556332bf0 | /virt/kvm/kvm_main.c:5152 | ./arch/x86/kvm/../../../virt/kvm/eventfd.c:1016 | 10452 | 0.437309 |
| 0xffffffff811ae0d1 | 0xffff8856e617ce84 | kernel/sched/fair.c:15198 | PO: ./kernel/sched/topology.c:1662 | 5422 | 0.410700 |
| ... | ... | ... | ... | ... | ... |
| 0xffffffff81366382 | 0xffff88559eb65f60 | ./mm/gup.c:2174 | ./arch/x86/mm/pgtable.c:30 | 1 | 0.000013 |
| 0xffffffff81366382 | 0xffff88559eb65f80 | ./mm/gup.c:2174 | ./arch/x86/mm/pgtable.c:30 | 1 | 0.000010 |
| 0xffffffff81366382 | 0xffff88559eb88620 | ./mm/gup.c:2174 | ./arch/x86/mm/pgtable.c:30 | 1 | 0.000008 |
| 0xffffffff81366382 | 0xffff88559eb88800 | ./mm/gup.c:2174 | ./arch/x86/mm/pgtable.c:30 | 1 | 0.000005 |
| 0xffffffff81366382 | 0xffff88559eb32110 | ./mm/gup.c:2174 | ./arch/x86/mm/pgtable.c:30 | 1 | 0.000003 |

```
1613 static int copy_signal(unsigned long clone_flags, struct task_struct *tsk)
1614 {
1615         struct signal_struct *sig;
1616
1617         if (clone_flags & CLONE_THREAD)
1618                 return 0;
1619
1620         sig = kmem_cache_zalloc(signal_cachep, GFP_KERNEL | GFP_NONSENSITIVE );
1621         tsk->signal = sig;
1622         if (!sig)
NORMAL  kernel/fork.c                                                    54% ☰
```

# Sorting by memory allocation

| variable | count | CDF |
| --- | --- | --- |
| <Unknown alloc> | 174907 | 1.000000 |
| PO: ./kernel/fork.c:241 | 56949 | 0.554715 |
| PO: ./kernel/fork.c:1620 | 52524 | 0.409732 |
| PO: ./kernel/fork.c:165 | 12411 | 0.276015 |
| ./kernel/events/core.c:10196 | 11082 | 0.244418 |
| ... | ... | ... |
| ./kernel/kthread.c:215 | 1 | 0.000013 |

| ip | address | function | variable | count |
| --- | --- | --- | --- | --- |
| 0xffffffff813bdbf1 | 0xffff88556abc1398 | ./mm/memcontrol.c:886 | <Unknown alloc> | 140972 |
| 0xffffffff811de7c3 | 0xffff885554673a20 | el/locking/lockdep.c:3311 | <Unknown alloc> | 28769 |
| 0xffffffff8119e82f | 0xffff88b684947f10 | /kernel/sched/fair.c:9254 | <Unknown alloc> | 4856 |
| 0xffffffff81366275 | 0xffff88bfdff660a8 | ./mm/gup.c:2174 | <Unknown alloc> | 62 |
| 0xffffffff8134ba16 | 0xffff88603fffe908 | ./mm/mmzone.c:68 | <Unknown alloc> | 30 |
| ... | ... | ... | ... | ... |
| 0xffffffffc00361f5 | 0xffff8856cf70c540 | net/google/gq/gq_tx.c:290 | <Unknown alloc> | 1 |
| 0xffffffffc00361f5 | 0xffff8856cf70c8a0 | net/google/gq/gq_tx.c:290 | <Unknown alloc> | 1 |
| 0xffffffffc00361f5 | 0xffff8856cf70d008 | net/google/gq/gq_tx.c:290 | <Unknown alloc> | 1 |
| 0xffffffffc003757c | 0xffff8856ce5ef9ff | net/google/gq/gq_rx.c:551 | <Unknown alloc> | 1 |
| 0xffffffff81b57cc6 | 0x7d1c7ffb43bd | x86/lib/copy_user_64.S:66 | <Unknown alloc> | 1 |

What to do with unknown allocations?

# Ratio of ASI-exits/VMEXIT's

SPECCPU-2006, perlbench_r, partial run

```
[235 rows x 2 columns]
VCPU 0: Time 199.85 seconds, asi/vm exits = 84898 / 5071891 = 1.67 %
VCPU 1: Time 199.85 seconds, asi/vm exits = 65118 / 125786 = 51.77 %
VCPU 2: Time 101.45 seconds, asi/vm exits = 71434 / 153760 = 46.46 %
VCPU 3: Time 101.49 seconds, asi/vm exits = 65966 / 271169 = 24.33 %
VCPU 4: Time 101.50 seconds, asi/vm exits = 68709 / 291743 = 23.55 %
VCPU 5: Time 101.50 seconds, asi/vm exits = 67701 / 125661 = 53.88 %
VCPU 6: Time 101.44 seconds, asi/vm exits = 64486 / 440259 = 14.65 %
VCPU 7: Time 101.50 seconds, asi/vm exits = 65211 / 196985 = 33.10 %
VCPU 8: Time 101.43 seconds, asi/vm exits = 113477 / 646273 = 17.56 %
VCPU 9: Time 101.50 seconds, asi/vm exits = 381747 / 674055 = 56.63 %
VCPU 10: Time 101.48 seconds, asi/vm exits = 64326 / 259593 = 24.78 %
VCPU 11: Time 101.44 seconds, asi/vm exits = 74550 / 148033 = 50.36 %
VCPU 12: Time 101.49 seconds, asi/vm exits = 69588 / 130233 = 53.43 %
VCPU 13: Time 101.50 seconds, asi/vm exits = 68766 / 283730 = 24.24 %
VCPU 14: Time 101.47 seconds, asi/vm exits = 65379 / 125270 = 52.19 %
VCPU 15: Time 101.48 seconds, asi/vm exits = 69624 / 137417 = 50.67 %
```