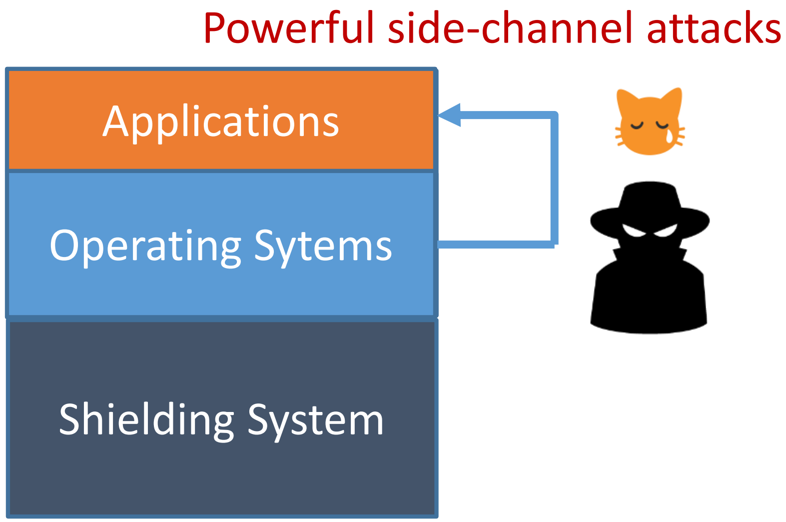
SHIELDING SOFTWARE FROM PRIVILEGED SIDE-CHANNEL ATTACKS

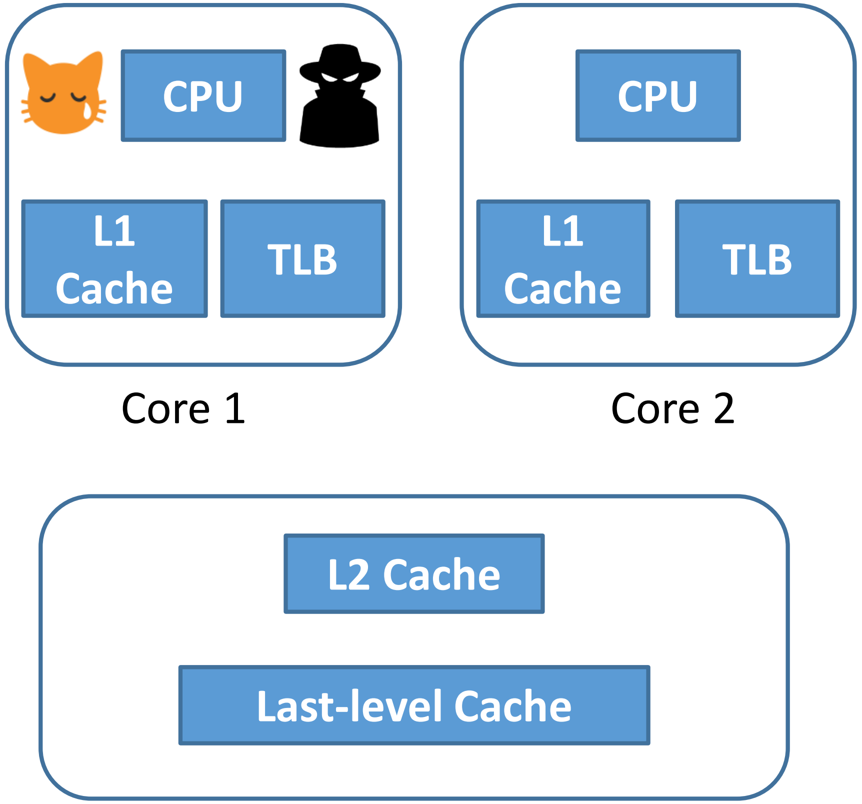
Xiaowan Dong	**University of Rochester**
Zhuojia Shen	University of Rochester
John Criswell	University of Rochester
Alan Cox	Rice University
Sandhya Dwarkadas	University of Rochester

OS-Launched Side-Channel Attacks



- Applications assume OS is secure, however...
- OS can be compromised
  - Buffer overflows, information leak
- Shielding systems like Intel SGX can protect confidential application data from direct corruption
- A compromised OS can still launch side-channel attacks

OS-Launched Side-Channel Attacks



- Exacerbate existing side channels
  - Infer the victim's behavior based on shared architectural states (*caches, TLBs*)
  - Control system events to alleviate noise
- Introduce new side channels
  - Trace page faults
  - Monitor page table updates

Why Shielding Systems Don't Help

- Shielding systems are supposed to protect confidential application data from compromised OS, however…
- Page table is still managed by untrusted OS
- The OS and other untrusted applications still share architectural states with applications needing protection (Caches, TLB, etc.)

We focus on defending against *page table side-channel* and *Last-level cache (LLC) side-channel* attacks from the OS kernel

Outline

- Examples of page table and cache side-channel attacks
- Background on Virtual Ghost
- Apparition
- Performance evaluation

Outline

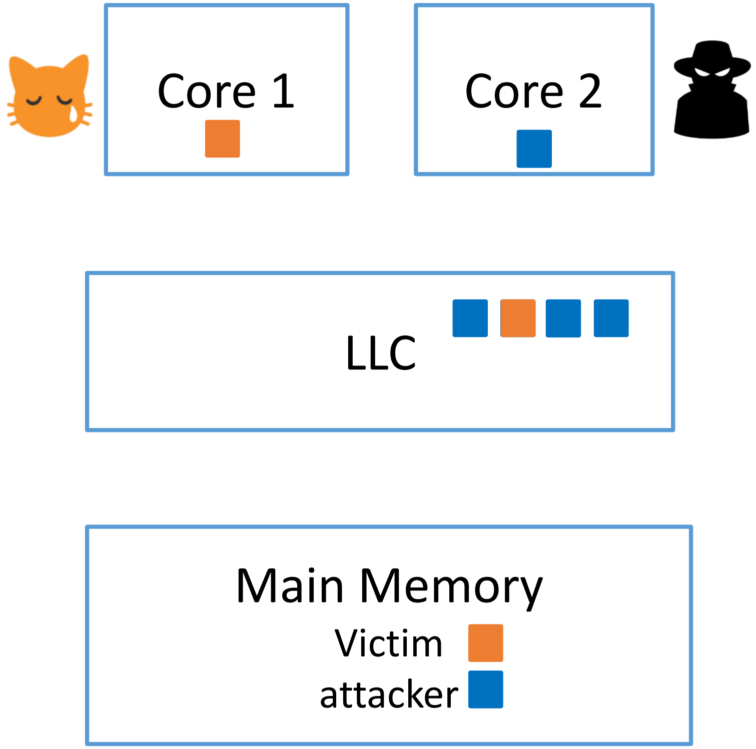
- **Examples of page table and cache side-channel attacks**
- Background on Virtual Ghost
- Apparition
- Performance evaluation

Page Table Side Channels

- Infer victim's memory access behavior
- Tracing page faults
  - Trigger page fault on every memory access
    - Requires page table modification
  - Can be used to recover entire secret document [1]
- Scanning ACCESS/DIRTY bit of page table entries
  - Monitor first memory read/write
    - First memory read/write sets ACCESS/DIRTY bit
    - Requires page table reads

[1] Yuanzhong Xu, Weidong Cui, and Marcus Peinado. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. Oakland. 2015.

LLC Side Channel: Prime + Probe Attack



- Attacker Infers the cache line accessed by the victim
- **Prime:** access ■
- **Idle:** while the victim accesses ■
- **Probe:** access ■ again
  - If latency is longer, the victim has replaced ■ with ■
- Flush + Reload has a similar rationale

We Need to Prevent Compromised OS from Reading or Writing…

- Confidential application data
- Page tables containing translations for confidential application data
- Cache lines of confidential application data

We leverage a shielding system called Virtual Ghost that already

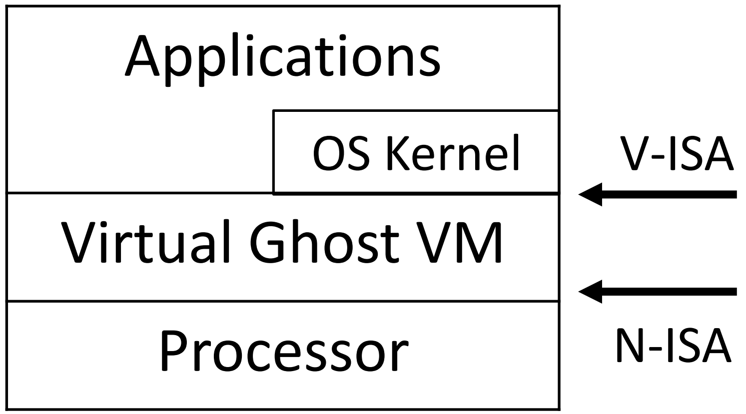
- Prevents OS from reading and writing confidential application data
- Controls how OS configures page table

Outline

- Examples of page table and cache side-channel attacks
- **Background on Virtual Ghost**
- Apparition
- Performance evaluation

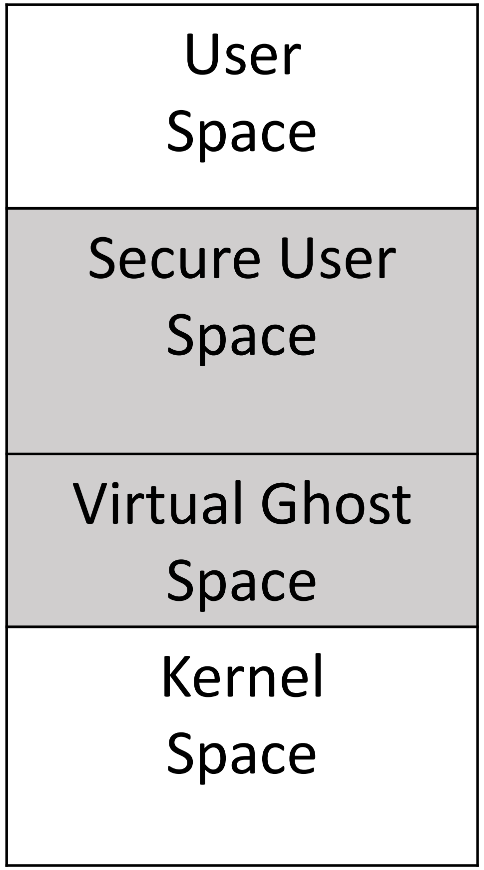
Background on Virtual Ghost

- A compiler-based virtual machine to protect application data from OS kernels
- Ports OS to virtual instruction set (V-ISA)
- Uses software fault isolation
- Forces OS kernel to invoke specific instructions to
  - Manipulate program state (e.g., context switch)
  - Configure hardware state (e.g., MMU)
- *Does not mitigate side-channel attacks*



Protected Memory Regions

- OS cannot access protected memory regions
  - Secure user space
    - Private to each application
  - Virtual Ghost space
    - Only accessible to Virtual Ghost VM
    - Used for saving internal data structures



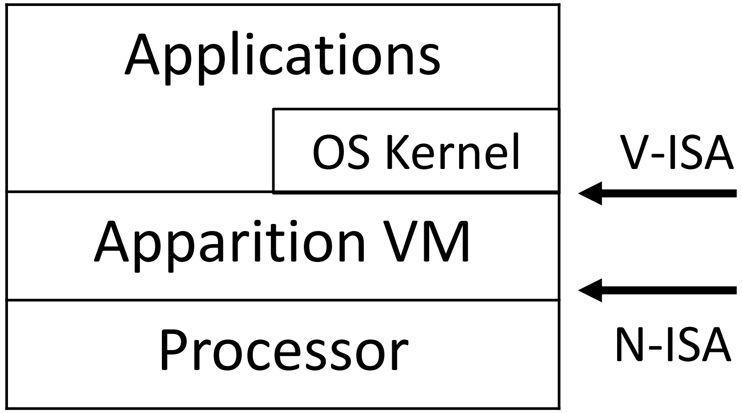
Virtual address space

Outline

- Examples of page table and cache side-channel attacks
- Background on Virtual Ghost
- **Apparition**
- Performance evaluation

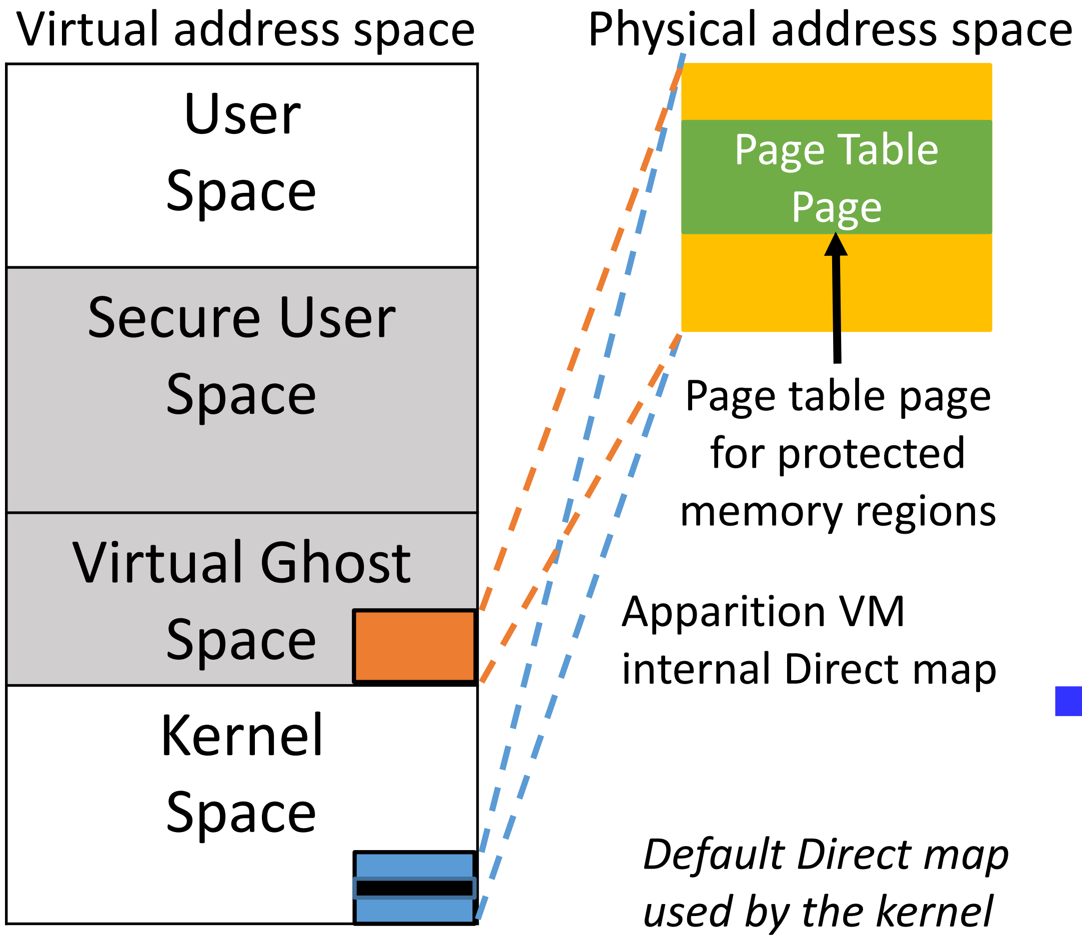
Apparition

- Virtual Ghost + page table and LLC side-channel defenses
- Control native code generation of the kernel
  - Ensure the kernel is instrumented



PAGE TABLE SIDE-CHANNEL DEFENSES

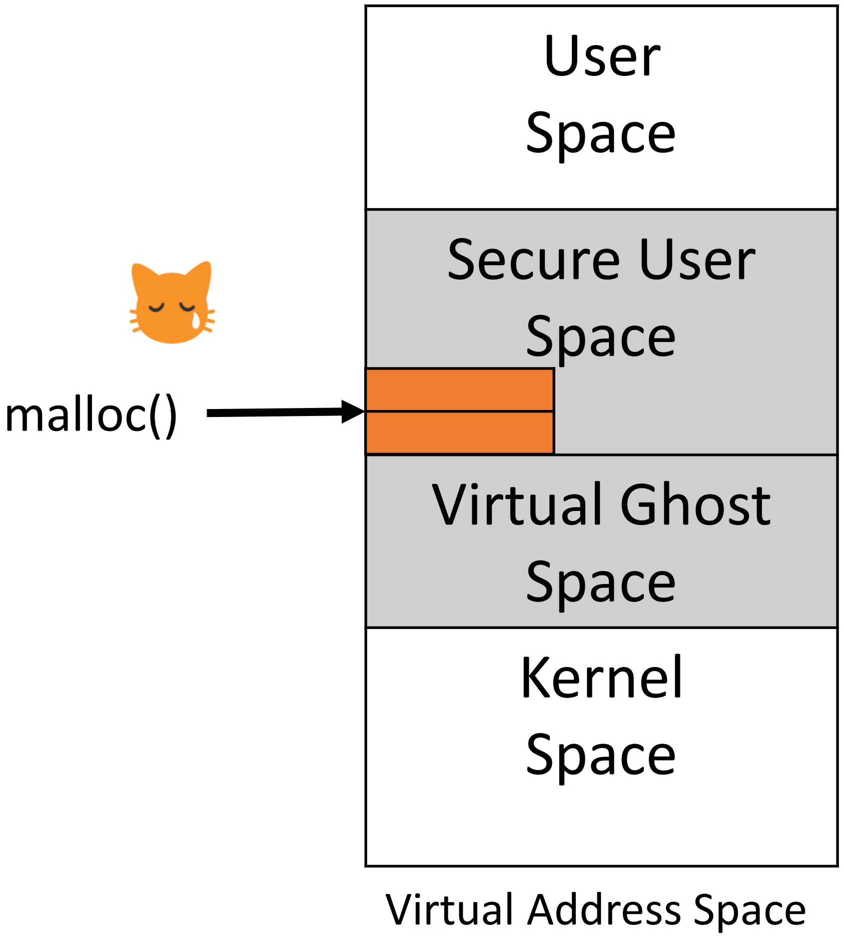
Page Table Side-Channel Defenses



- Direct map: a range of virtual memory mapping the entire physical memory as a single block
- Page table pages normally accessed via direct map
- Prevent OS from reading or writing the page table of the protected memory regions

➡ Remove the entry mapping the page table page from the kernel's direct map

Paging Side Channels Defenses

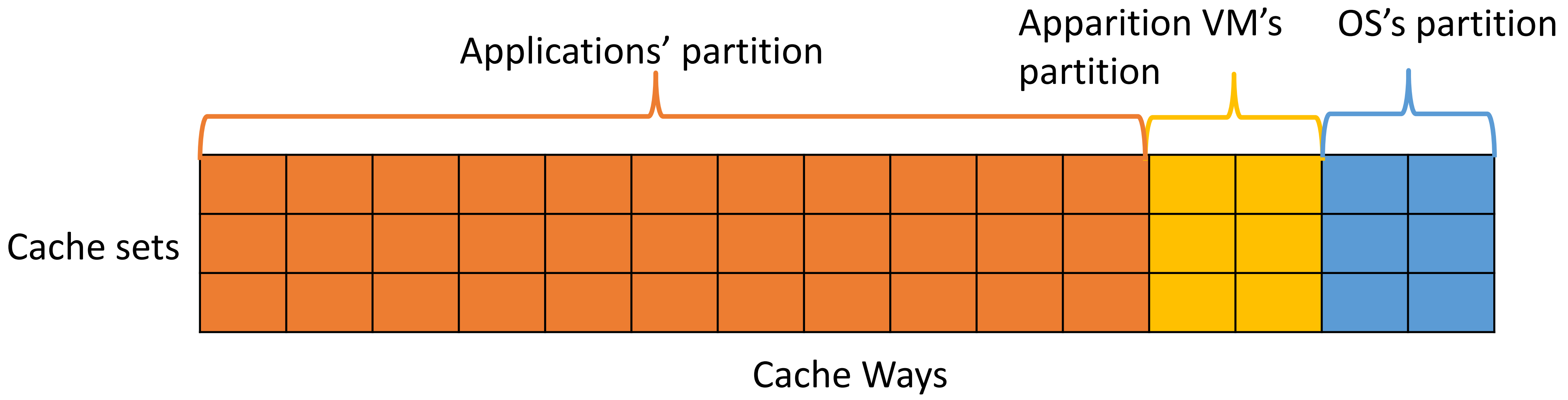


- *Lazy memory allocation*
  - OS maps the frame to the page when the application first reads or writes the page
- Side channel
  - Reveals paging behavior of the victim
- Defenses
  - Apparition VM manages secure user space memory allocation instead of OS
  - Map physical frames upon allocation rather than at access time

LLC
SIDE-CHANNEL
DEFENSES

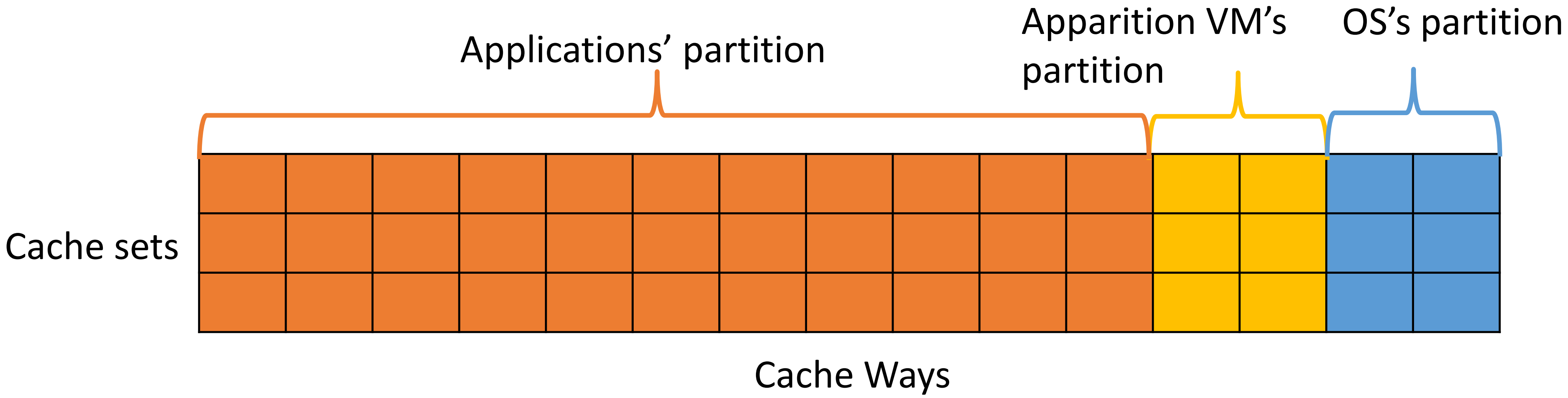
Defenses against LLC Side Channels

- Partition LLC to isolate applications from OS
- Assign different cache partitions to OS, Apparition VM, and applications needing protection



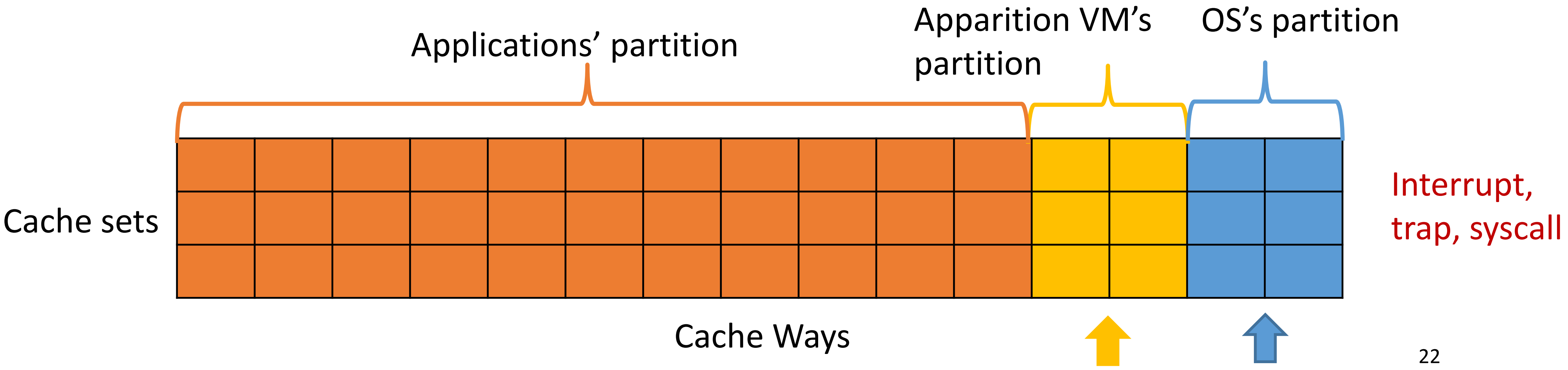
Intel Cache Allocation Technology

- Hardware feature that partitions LLC ways into subsets of smaller associativities
- Code can only *evict* cache lines in its partition, but can *read* any part of the LLC (no isolation on reads)
- Secure user space cache lines are not readable by OS



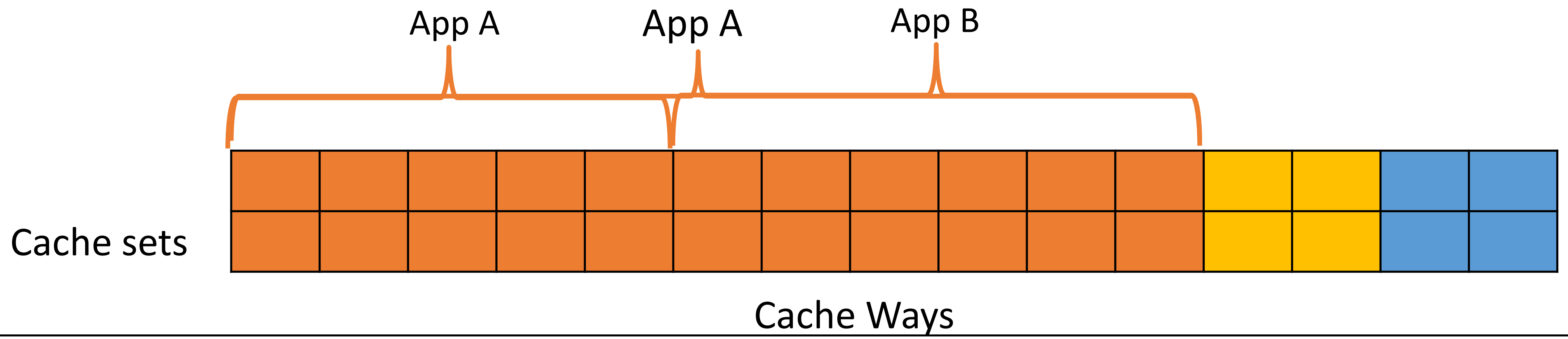
Cache Partitioning Configuration

- Apparition VM
  - configures cache partitioning at boot time
  - prevents the OS from reconfiguring the partitions via its virtual instruction set
  - switches to the corresponding cache partition based on the code running (application, Apparition VM, and OS)



Private Cache Partitions for Applications

- Each application needing protection has its own cache partition
- First assign one cache partition to the first application
- Then divide it when more applications are scheduled
- Hardware partitions are shared when they run out
  - Flush the cache over context switch between two applications sharing partitions



Spectre and Meltdown Attacks

- Apparition helps prevent information leak via LLC side channel
  - Mitigates LLC side-channel attacks

- Our HASP paper [1] presents SFI that mitigates Spectre variant 1 and Meltdown

[1] Xiaowan Dong, Zhuojia Shen, John Criswell, Alan Cox, and Sandhya Dwarkadas. **Spectres, Virtual Ghosts, and Hardware Support.** In HASP '18.

Outline

- Examples of page table and cache side-channel attacks
- Background on Virtual Ghost
- Apparition
- **Performance evaluation**

Methodology

Experiment environment

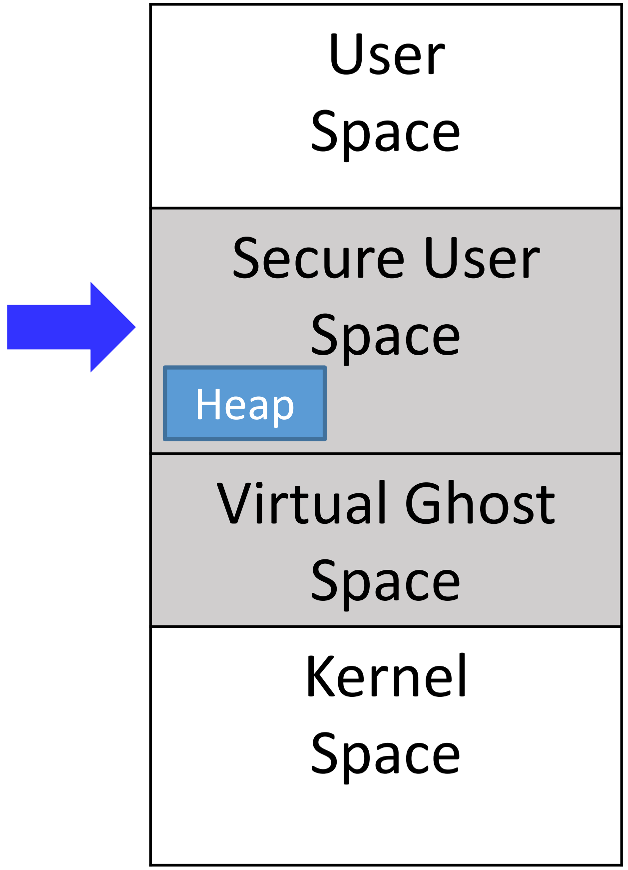
- 3.4 GHz Intel i7-6700 hyperthreading quad-core processor
- 16-way 8 MB LLC
- 16 GB RAM
- 256 GB SSD
- FreeBSD 9.0 ported to Apparition

Applications

- Tested CPU-intensive, network-intensive and file-system benchmarks
- A microbenchmark (that randomly accesses a large array)
- OpenSSH client
- Bzip2
- GnuPG
- Clang
- Highlight the results of a subset of applications evaluated

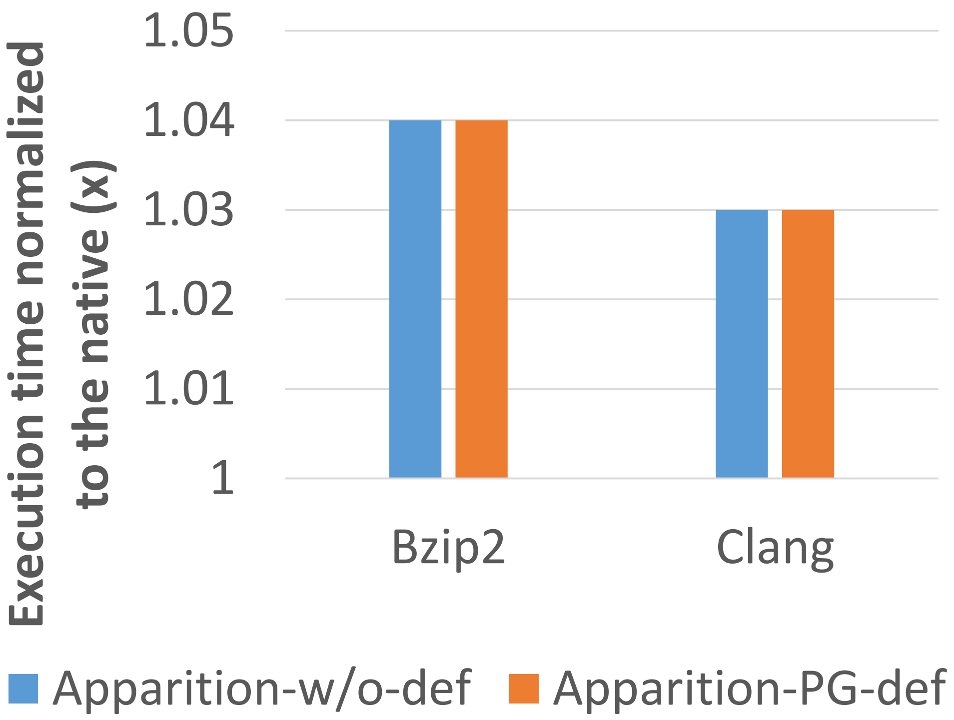
Methodology

- Bzip2:compress a 32 MB file
- Clang: compile gcc-smaller.c from SPEC CPU 2017
- GnuPG: cryptography program signing files of varying sizes
- All the applications put the heap in secure user space
  - We modified malloc() in *libc.so*



Virtual address space

Page Table Side-Channel Defenses Overheads



- *No additional overhead* on Bzip2 and Clang
- Disabling lazy memory allocation does not incur overhead
- Bzip2 and Clang access most of the heap allocated at runtime

Page Table Side-Channel Defenses Overheads

File Size	Apparition-w/o-def	Apparition-PG-def
1 KB	9.5 ms	23.7 ms
2 KB	9.5 ms	23.8 ms
…	x ms	(x + 14) ms
16 MB	386.2 ms	400.1 ms
32 MB	761.8 ms	776.1 ms

GnuPG signing files results.

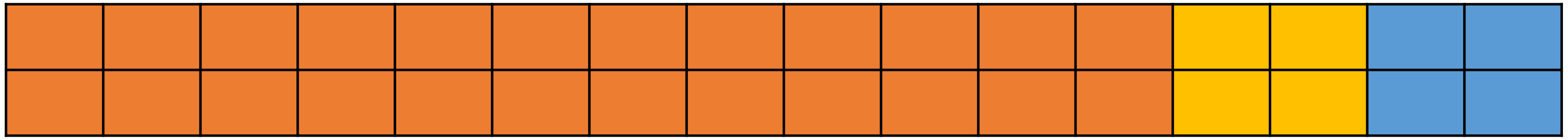
- A constant overhead of 14 ms due to disabling lazy memory allocation
- Additional cost of allocating and mapping 8 MB physical memory that is not accessed at runtime
  - Due to alignment issue of the first invocation of malloc()
- Overhead negligible as file size increases

LLC Side-Channel Defenses

- Simple approach: We statically partition the LLC into *three parts*
  - Our processor supports four cache partitions

- Experimentally determined cache partitioning that is close to the baseline performance

Application	12 ways
Apparition VM	2 ways
OS kernel	2 ways

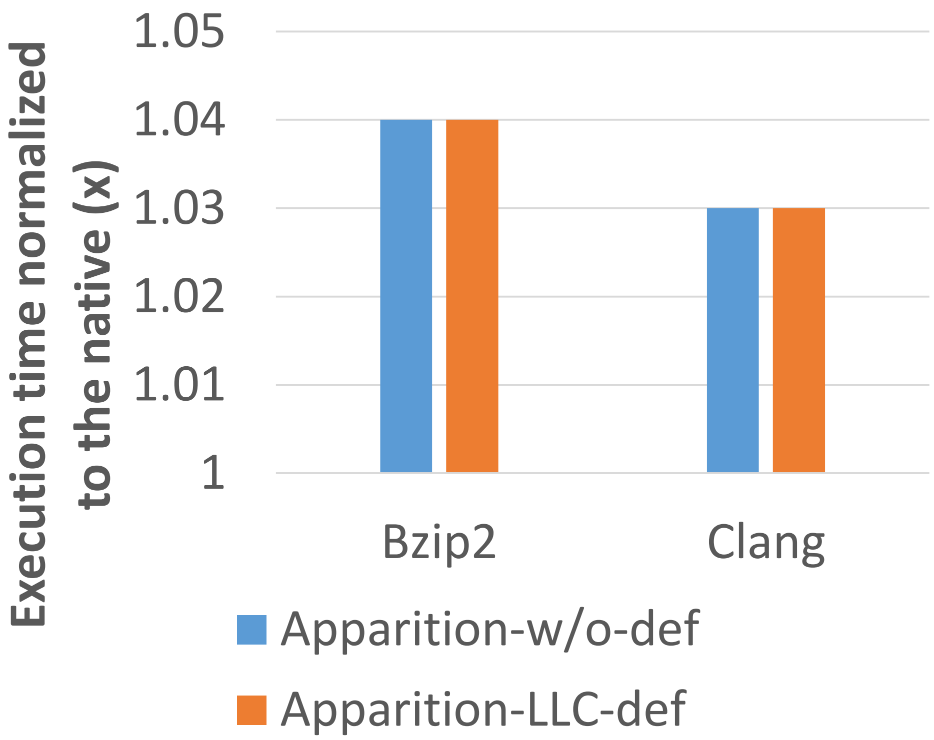


Applications' partition

Apparition VM's partition

OS's partition

LLC Side-Channel Defenses Overheads



- No additional overhead to Bzip2 and Clang

LLC Side-Channel Defenses Overheads

File Size	Apparition-w/o-defenses	Apparition-LLC-def
1 KB	9.5 ms	12.1 ms
2 KB	9.5 ms	12.1 ms
…	…	…
4 MB	103.9 ms	108.0 ms
8 MB	198.6 ms	203.6 ms
16 MB	386.2 ms	394.6 ms
32 MB	761.8 ms	776.6 ms

GnuPG signing files results.

- Switching among different LLC partitions incurs overhead
- Larger file size ⇒ more read/write syscalls ⇒ larger cache partition switching overhead
- For 8 MB to 32 MB files, the overhead is negligible (1.05x on average)

Conclusion

- Compromised OS is powerful enough to exacerbate existing side channels and introduce new side channels
- A compiler-based approach like Virtual Ghost can be leveraged to mitigate OS-launched page table and LLC side-channel attacks
- Apparition defends against page table and LLC side-channel attacks with low overhead (1% to 18%)