Hodor: Intra-Process Isolation for High-Throughput Data Plane Libraries

Mohammad Hedayati[1], Spyridoula Gravani[1], Ethan Johnson[1], John Criswell[1], Michael L. Scott[1], Kai Shen[2], Mike Marty[2]

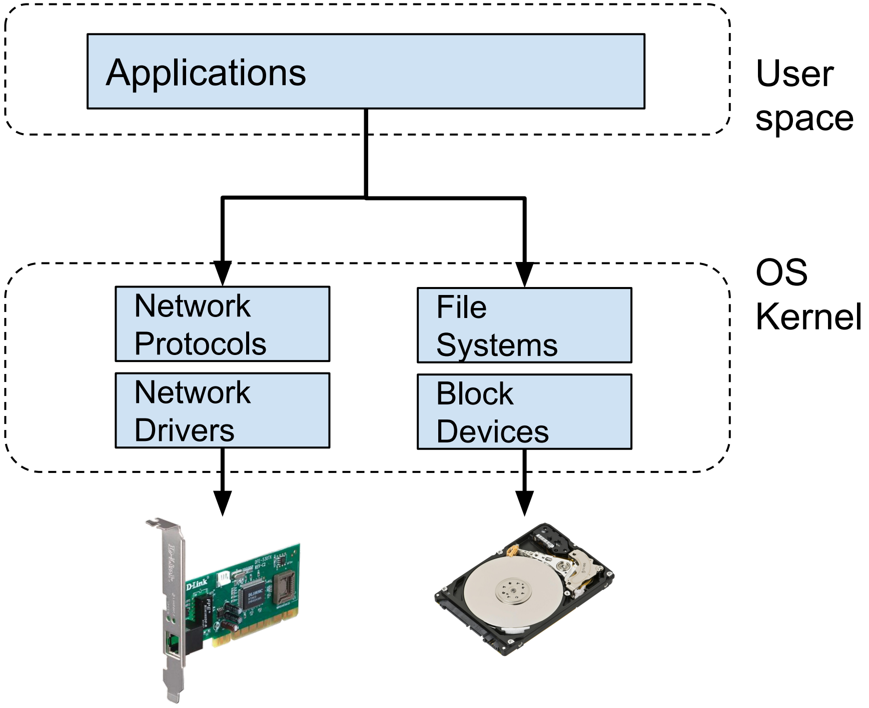
[1]University of Rochester
[2]Google

2019 USENIX Annual Technical Conference

Kernel I/O

- Enables sharing
- Provides guarantees
  - Fairness
  - Recovery
  - Security

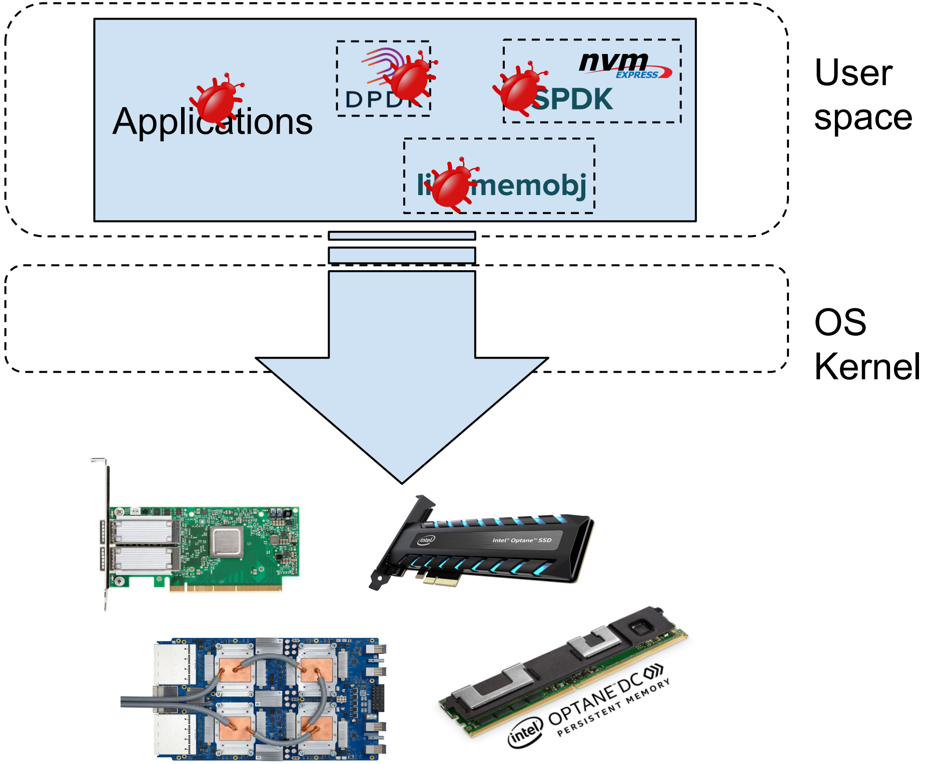
- All without needing to trust



Kernel-bypass I/O

- Pros:
  - Lower latency
  - Rapid development
  - Specialization

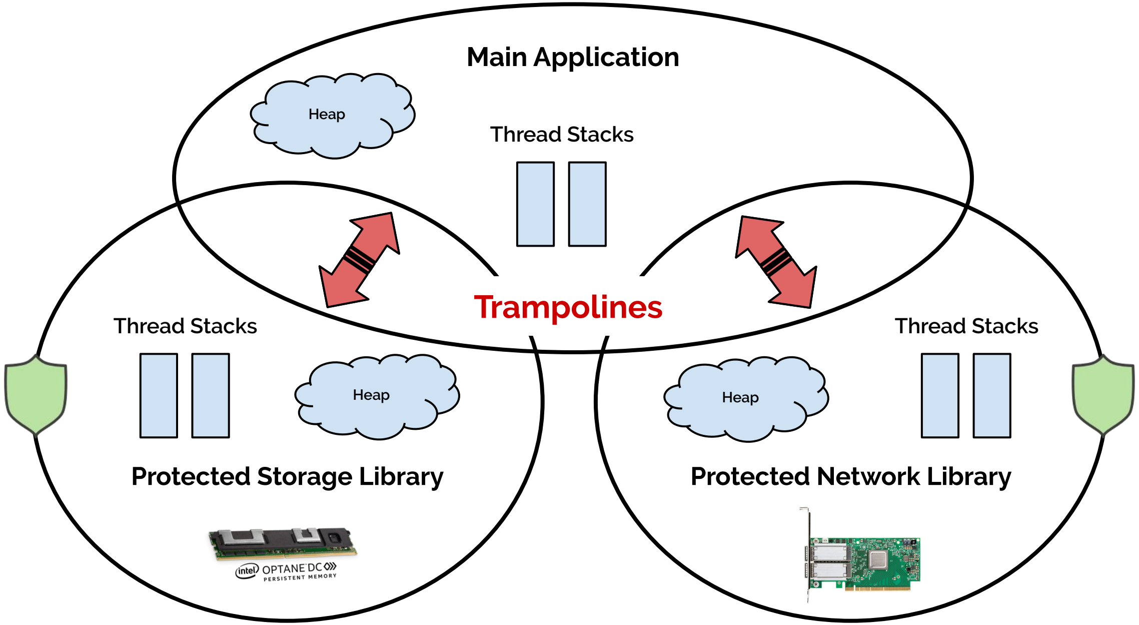
- Cons:
  - No guarantees
  - Hard to multiplex



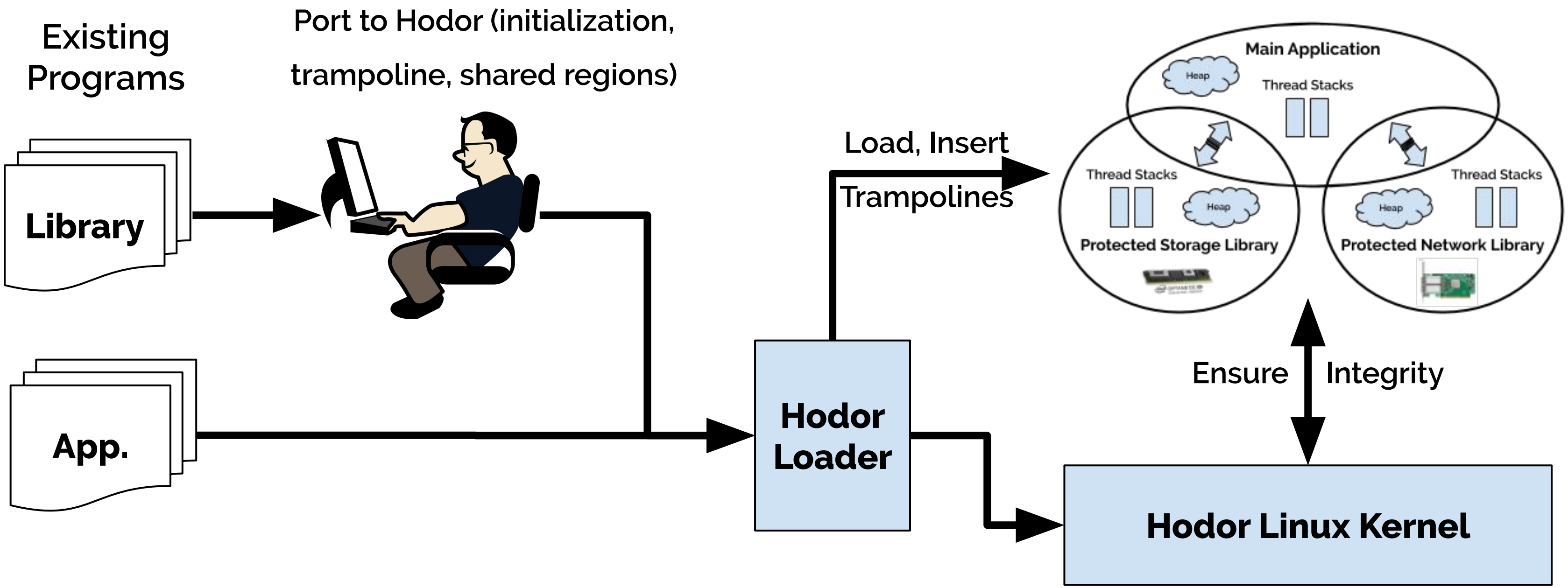
Overview

- Motivation
- Design of Hodor
- Fast Memory Isolation
- Evaluation
- Conclusion

Protected Library



Hodor



Existing Programs

Port to Hodor (initialization, trampoline, shared regions)

Library

App.

Hodor Loader

Load, Insert Trampolines

Ensure Integrity

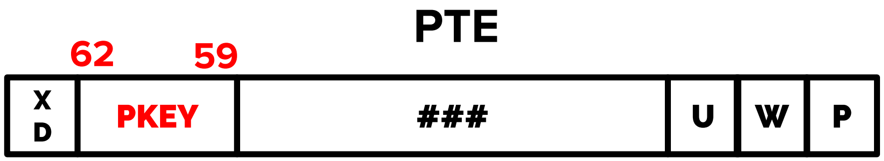
Hodor Linux Kernel

Overview

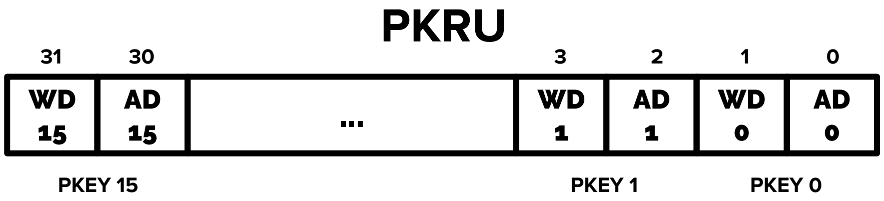
- Motivation
- Design of Hodor
- Fast Memory Isolation
- Evaluation
- Conclusion

Intel PKU

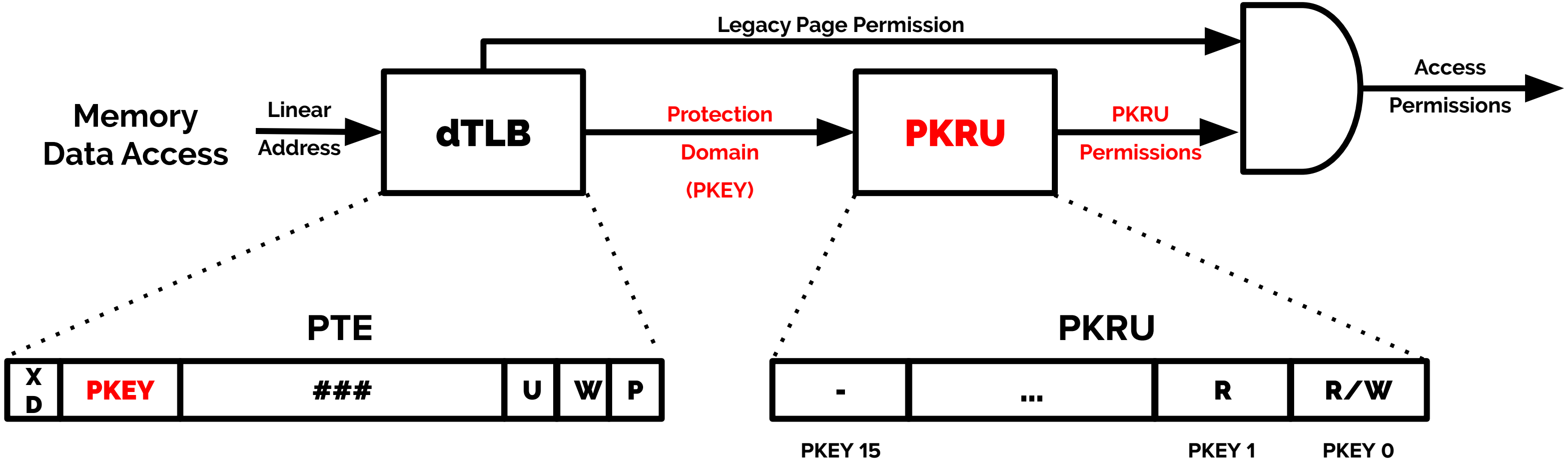
- Protection Keys for User-Space (a.k.a. MPK)
- Introduced in Skylake-SP



- 32-bit PKRU register (Access/Write Disable)
- `WRPKRU/RDPKRU`



Intel PKU



PTE

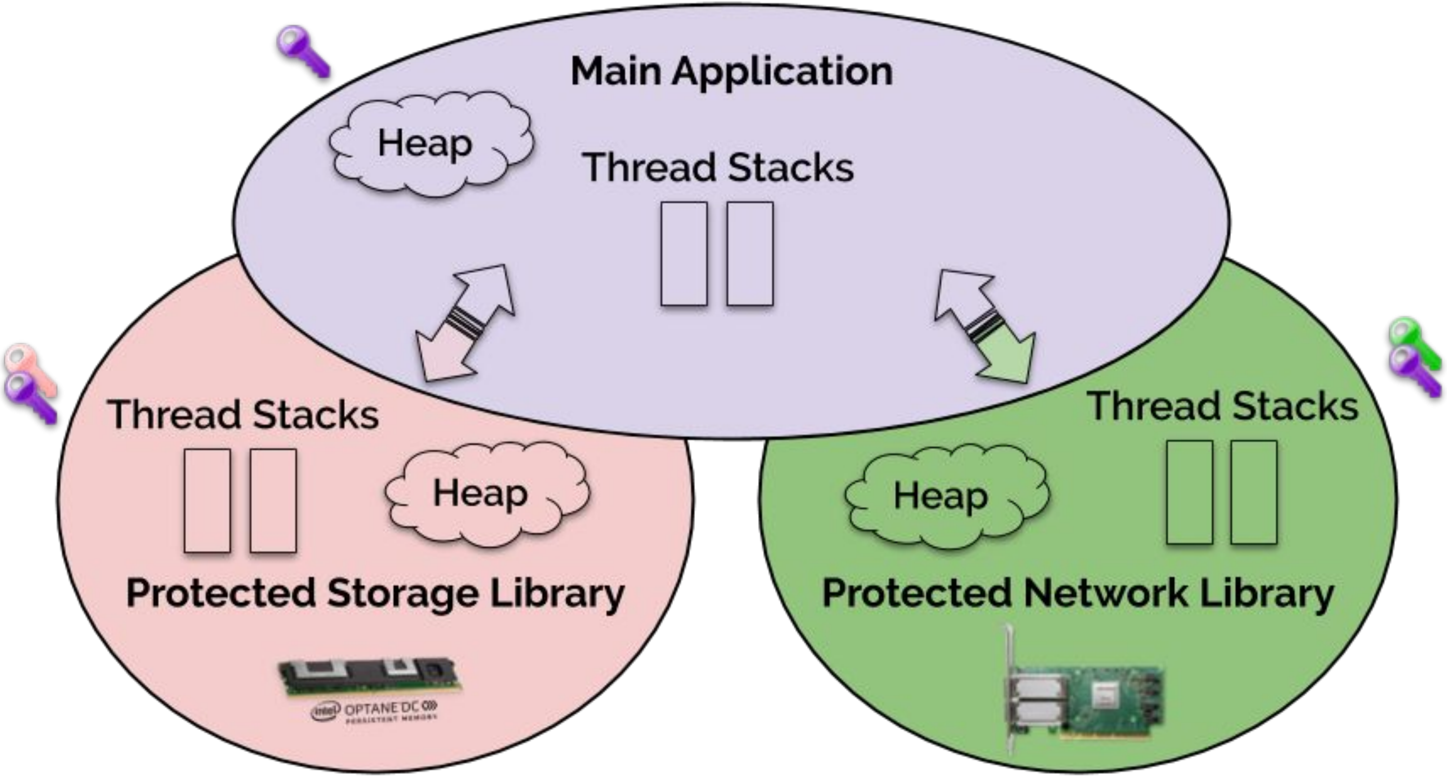
XD	PKEY	###	U	W	P

PKRU

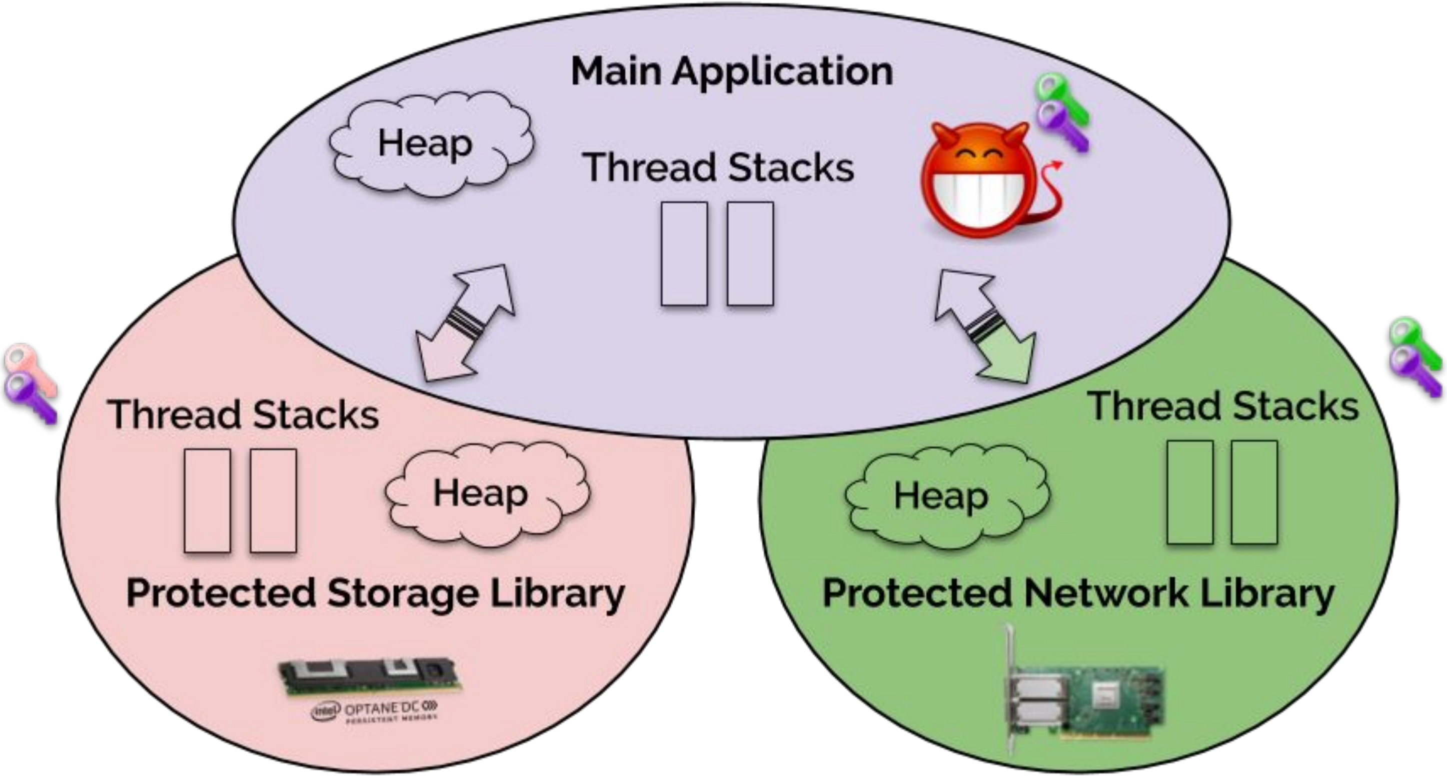
PKEY 15		PKEY 1	PKEY 0
-	...	R	R/W

based on a figure from: https://ubm.io/2YjGvFE

Hodor: Memory Isolation



Hodor: Memory Isolation



Hodor: Vetting WRPKRUS

- Inspect executable regions
  - Load (by Hodor loader)
  - W→X change (by Hodor kernel at run-time)
- Look for WRPKRU (0f 01 ef) instances

glibc-devel-static-2.27-alt9.x86_64

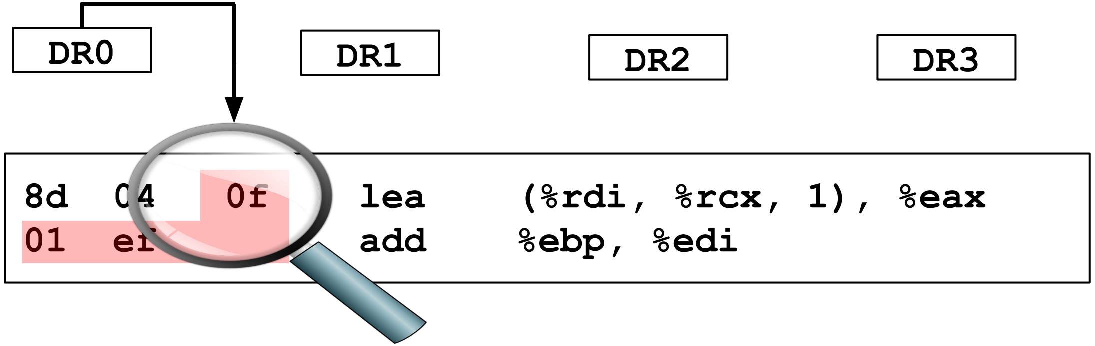
```
f7 d2       not    %edx
21 d0       and    %edx,%eax
44 89 c2    mov    %r8d,%edx
09 f0       or     %esi,%eax
0f 01 ef    wrpkru
31 c0       xor    %eax,%eax
```

blender-2.79b-7.fc29.x86_64

```
8d  04  0f    lea   (%rdi, %rcx, 1), %eax
01  ef        add   %ebp, %edi
```

Hodor: Vetting WRPKRUs

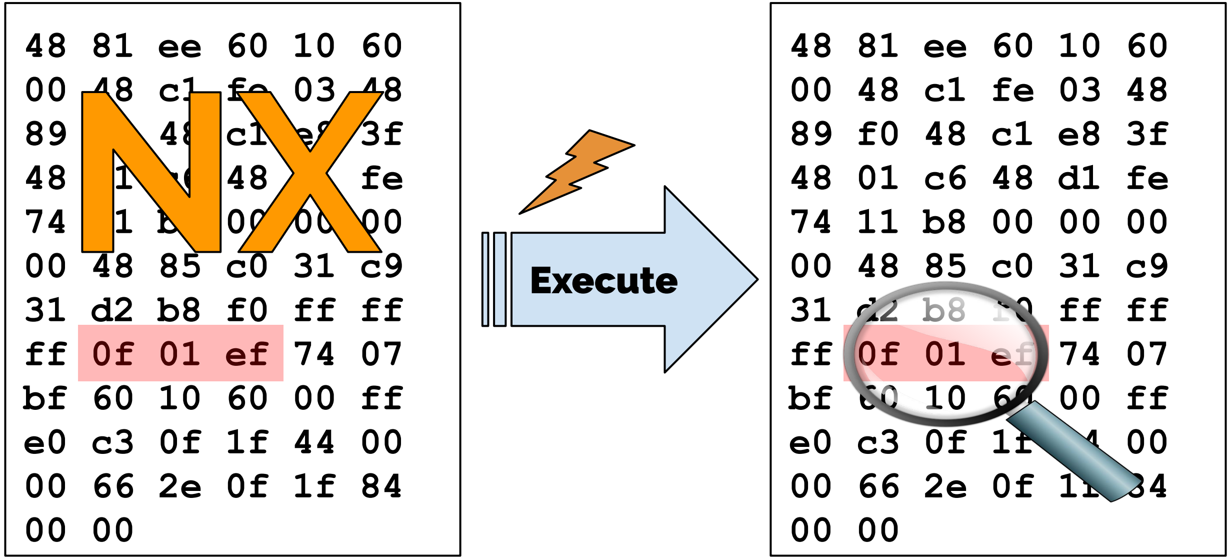
- Hardware watchpoints
  - DR# registers point to the beginning of illegal byte sequence
  - No spurious traps when correctly aligned execution runs past an implicit instance



```
8d  04  0f   lea    (%rdi, %rcx, 1), %eax
01  ef       add    %ebp, %edi
```

Hodor: Illegal `WRPKRU`s

- Limited hardware watchpoints
    - Only 4 on Intel Processors
    - HW watchpoints as **cache** for illegal sequences



```
48 81 ee 60 10 60
00 48 c1 fe 03 48
89 f0 48 c1 e8 3f
48 01 c6 48 d1 fe
74 11 b8 00 00 00
00 48 85 c0 31 c9
31 d2 b8 f0 ff ff
ff 0f 01 ef 74 07
bf 60 10 60 00 ff
e0 c3 0f 1f 44 00
00 66 2e 0f 1f 84
00 00
```

```
48 81 ee 60 10 60
00 48 c1 fe 03 48
89 f0 48 c1 e8 3f
48 01 c6 48 d1 fe
74 11 b8 00 00 00
00 48 85 c0 31 c9
31 d2 b8 f0 ff ff
ff 0f 01 ef 74 07
bf 60 10 60 00 ff
e0 c3 0f 1f 44 00
00 66 2e 0f 1f 84
00 00
```

Hodor: Vetting `WRPKRUS`

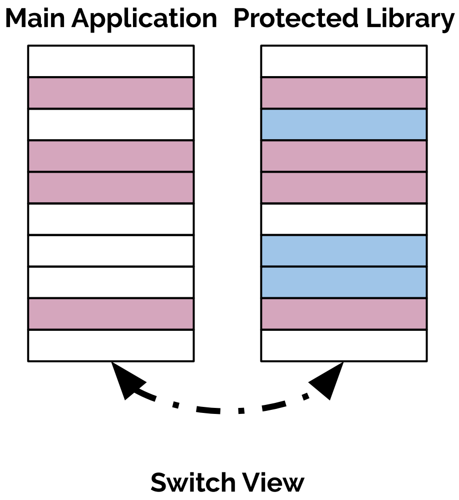
- Vetting cost
  - Implicit instances incur no run-time overhead
  - Explicit instances should use `pkey_set()`
  - No measurable overhead as long as:
    - #hot illegal seq. fewer than #hw watchpoints

- How often?
  - **58,273** rpm packages on Fedora 29 (108K executables)
  - Only 123 binaries with one or more illegal byte sequences
    - Only 2 (less than 0.02%) with more than 4

Alternative Memory Isolation

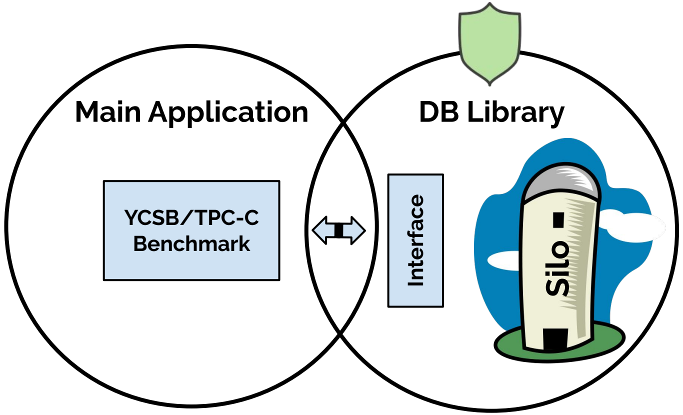
- Per-Domain Page-Table
  - Each mapping the view of a domain
  - Switch using system calls

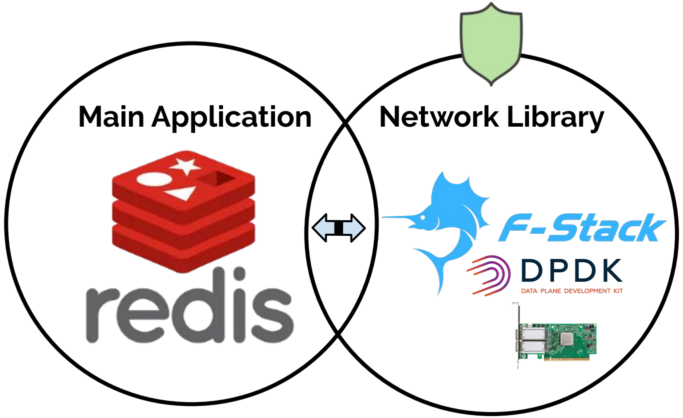
- Per-Domain Extended Page-Table
  - Requires running virtualized (in Intel VMX)
  - Switch using VMFUNC w/o causing VMEXIT



Evaluation: Applications

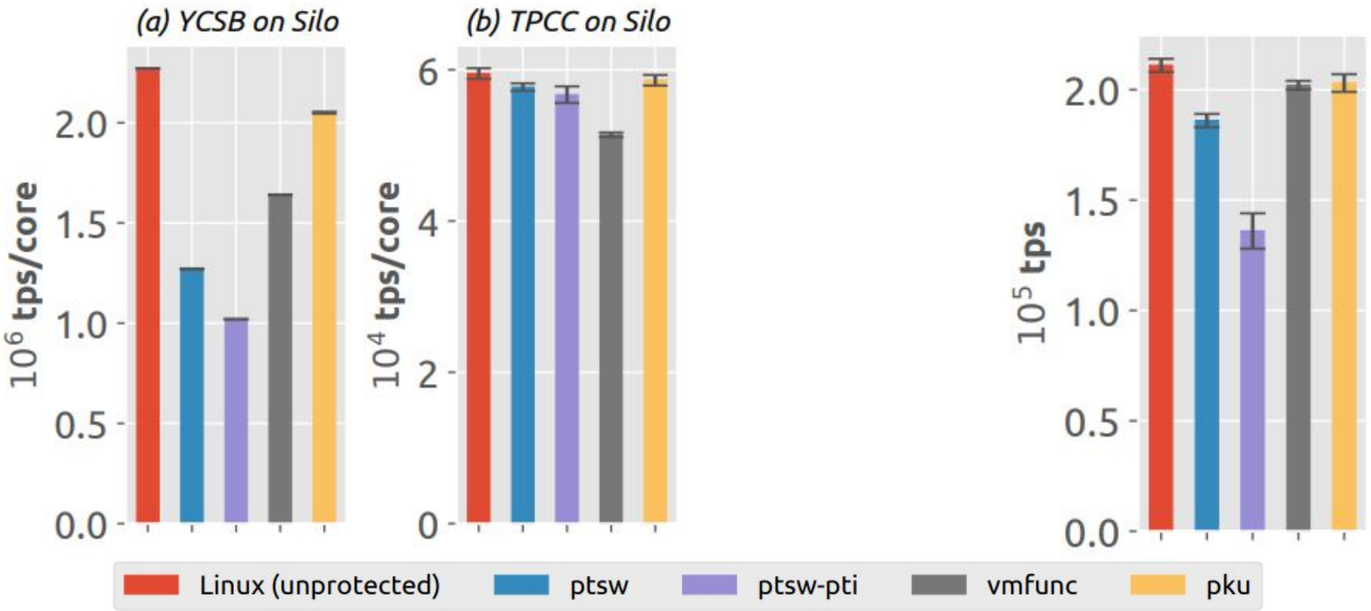
- Silo (in-memory database)
- Redis (kernel-bypass network TCP/IP stack)
- DPDK (kernel-bypass packet processing) -- in the paper





Evaluation: Applications

Silo **Redis**



ptsw: page table switching
vmfunc: EPT switching

ptsw-pti: page table switching w/ KPTI,
pku: using memory protection keys

Conclusion

Introduced:

- *Protected Libraries*: new OS abstraction for library isolation

Showed that:

- Intel PKU can be safely used to isolate protected libraries
- Doing so does not sacrifice performance
    - **90–98%** of unprotected throughput

See the paper: How multiple processes can share a protected library