IPADS
INSTITUTE OF PARALLEL AND DISTRIBUTED SYSTEMS

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Harmonizing Performance and Isolation in Microkernels with New Hardware

Zeyu Mi

IPADS, Shanghai Jiao Tong University

https://ipads.se.sjtu.edu.cn/pub/members/zeyu_mi

Microkernels are rising again

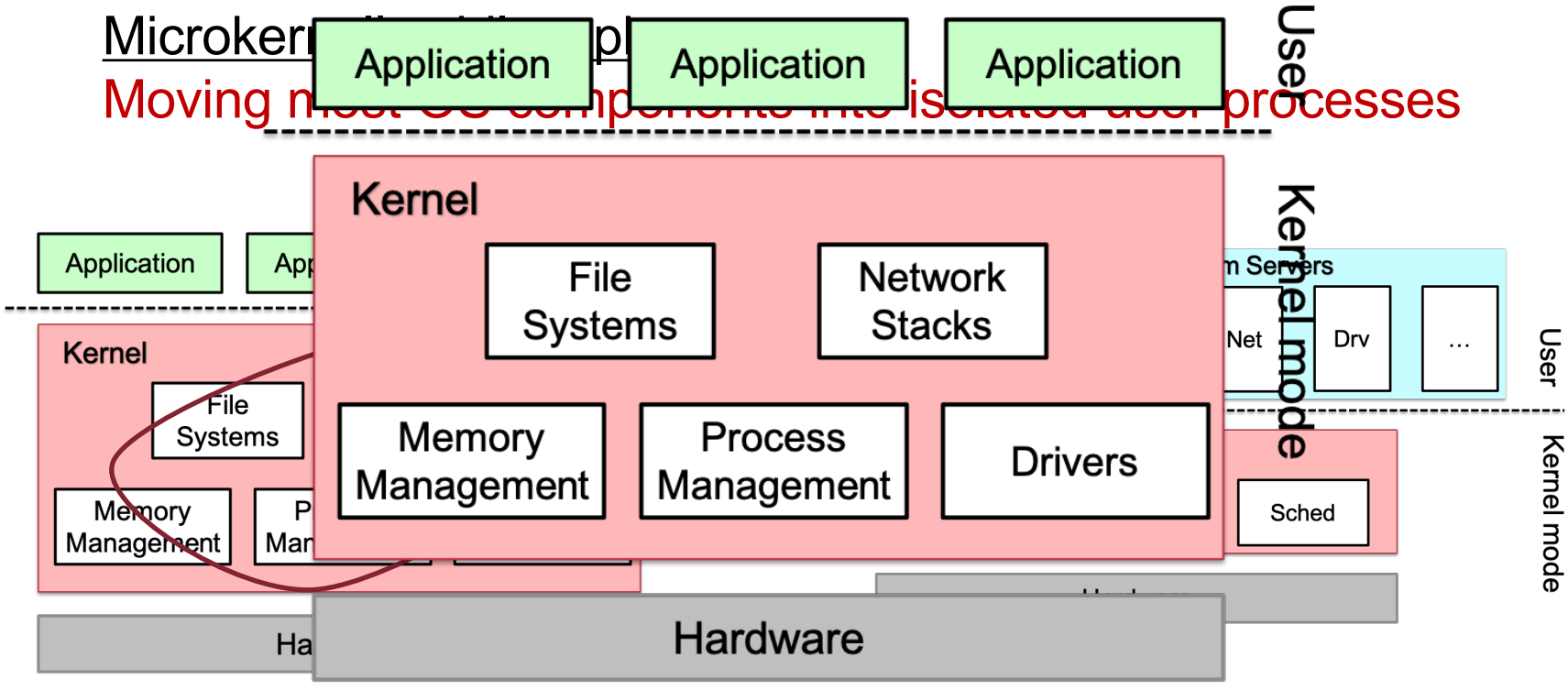
- Achieves good extensibility, security, and fault isolation
- Succeeds in safety-critical scenarios (Airplane, Car)
- For more general-purpose applications

HarmonyOS 鸿蒙
基于微内核的全场景分布式OS

QNX SOFTWARE SYSTEMS

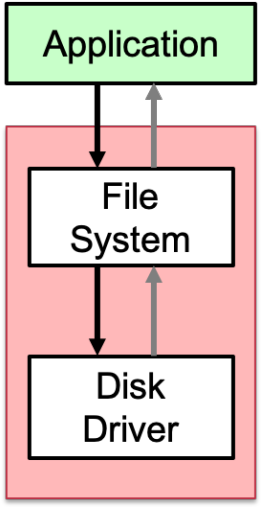
Fuchsia by Google

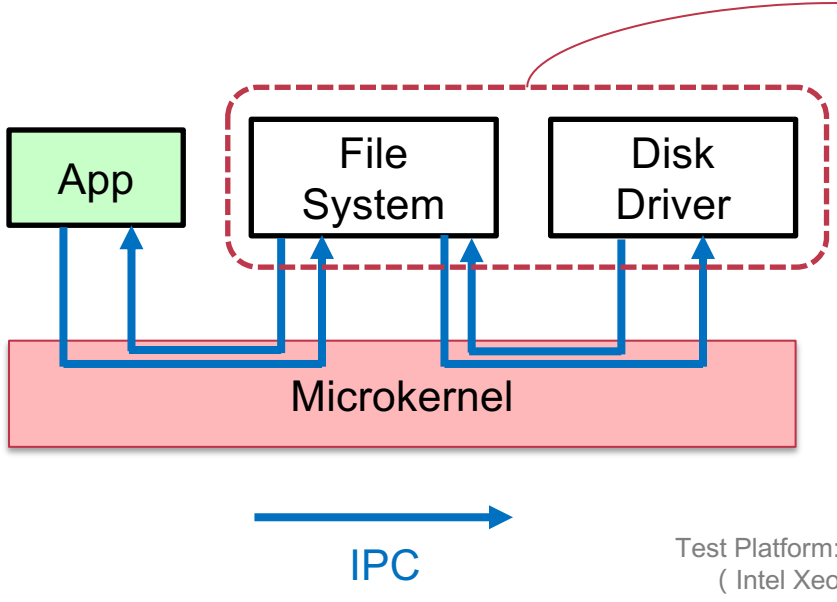
Monolithic Kernel and Microkernel

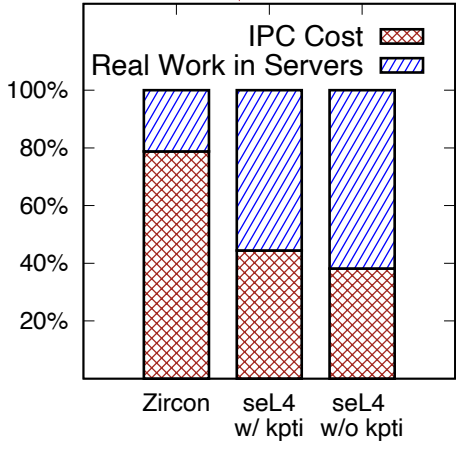


Performance and Isolation Trade-off

- Function Call -> Inter-Process Communication (IPC)
- Overhead of IPC : Direct cost and indirect cost



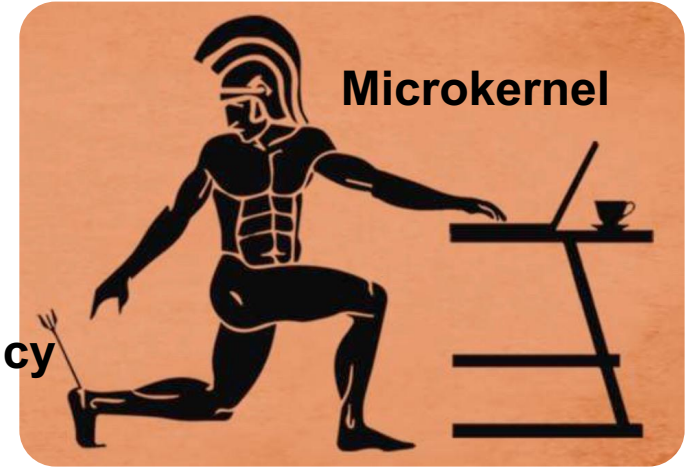




Test Platform: Dell PowerEdge R640
(Intel Xeon Gold 6138 CPU)

Goal: Both Ends

- **Harmonize the tension between Performance and Isolation in microkernels**
  - Reducing the IPC overhead
  - Maintaining the isolation guarantee



IPC IS THE ACHILLE'S HEEL OF MICROKERNELS

Direct Cost

Each IPC has to involve the kernel

The involvement of kernel brings huge direct costs

	Cost
Syscall + Sysret	157 cycles
Two PT switches	*372 cycles*
Other logics	~150 cycles
Total	~680 cycles

Indirect Cost

IPC also causes indirect costs

The sources of indirect costs

- Pipeline
- Different levels caches
- TLB structures

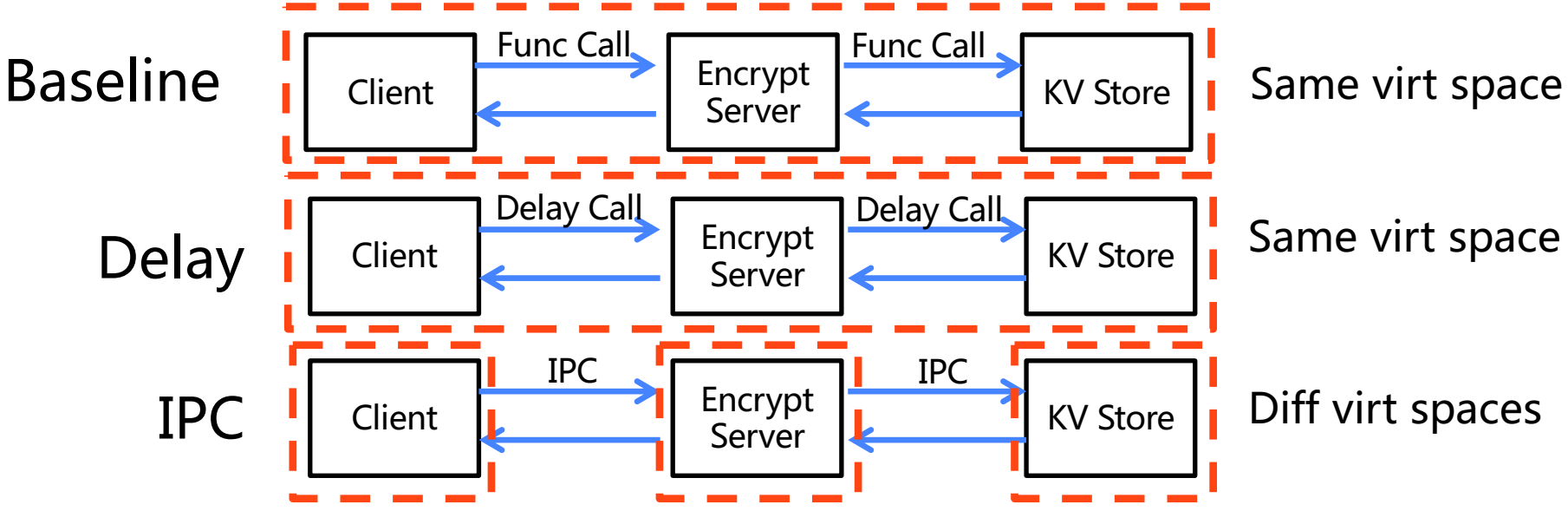
Indirect costs affect the performance of user processes

Indirect Cost

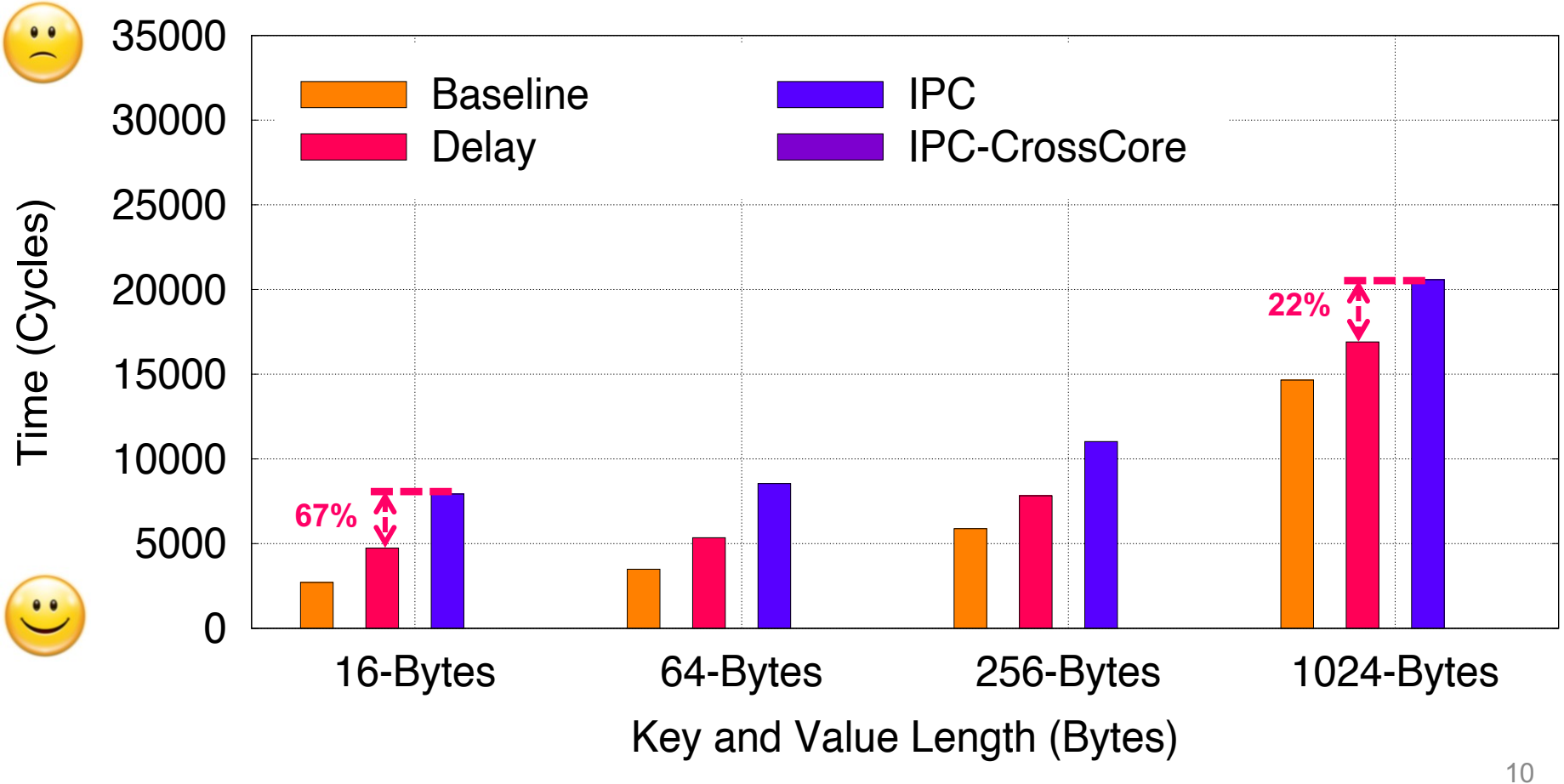
An experiment to show the indirect costs

Three processes: a client, an encryption server and a KV store

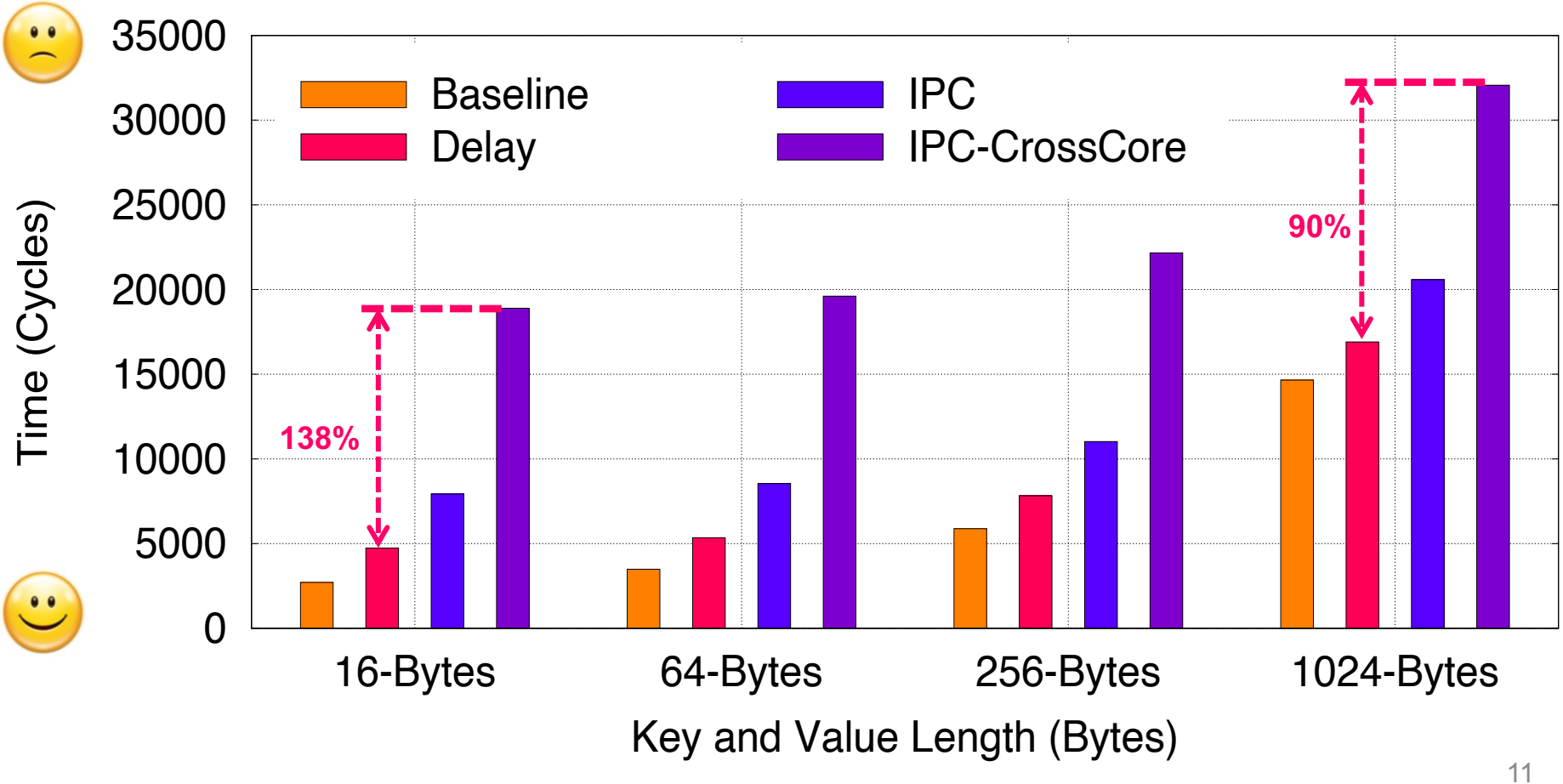
Three configurations:



Indirect Cost



Indirect Cost



Indirect Cost

Name	i-cache	d-cache	L2 cache	L3 cache	i-TLB	d-TLB
Baseline	15	10624	13237	43	8	17
Delay	15	10639	13258	43	9	19
IPC	696	27054	15974	44	11	7832

- **Affect architectural structures**
  - Measure architectural pollution using Intel PMU
  - Count events for doing 512 operations

Previous Solutions

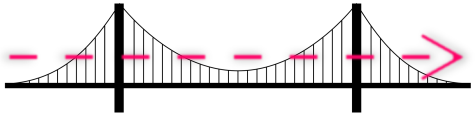
- **A long line of optimizations in the last 30 years**
  - LRPC (SOSP' 89)、L4 (SOSP' 95)、seL4 fastpath (SOSP' 09)、dIPC (EuroSys' 17) …
  - Try to mitigate the effect of kernel involvement: Bypassing scheduler, Reusing states, etc.
  - Most are software-based solutions and still have large overhead
- **But the relative latency of IPC gets larger when hardware becomes faster**

Latency	Monolithic Kernel	Microkernel	Ratio
30 years ago	21 µs	114 µs	**4.8 X**
Modern CPU	0.049 µs	0.5 µs	**10.2 X**

SkyBridge: Fast and Secure Inter-Process Communication for Microkernels

EuroSys 2019

System Overview



SkyBridge: a fast and secure IPC facility for microkernels.

1. Direct IPC path without the kernel.
2. Maintain the same level of security as the traditional IPC
3. Easily integrated into existing microkernels.

Performance improvement

- Mircobenchmarks: **1.49X** – **19.6X** speedup for IPC
- Real Apps: **81.9%** – **9.59X** improvement for SQLite 3.0

EPT Switching: VMFUNC

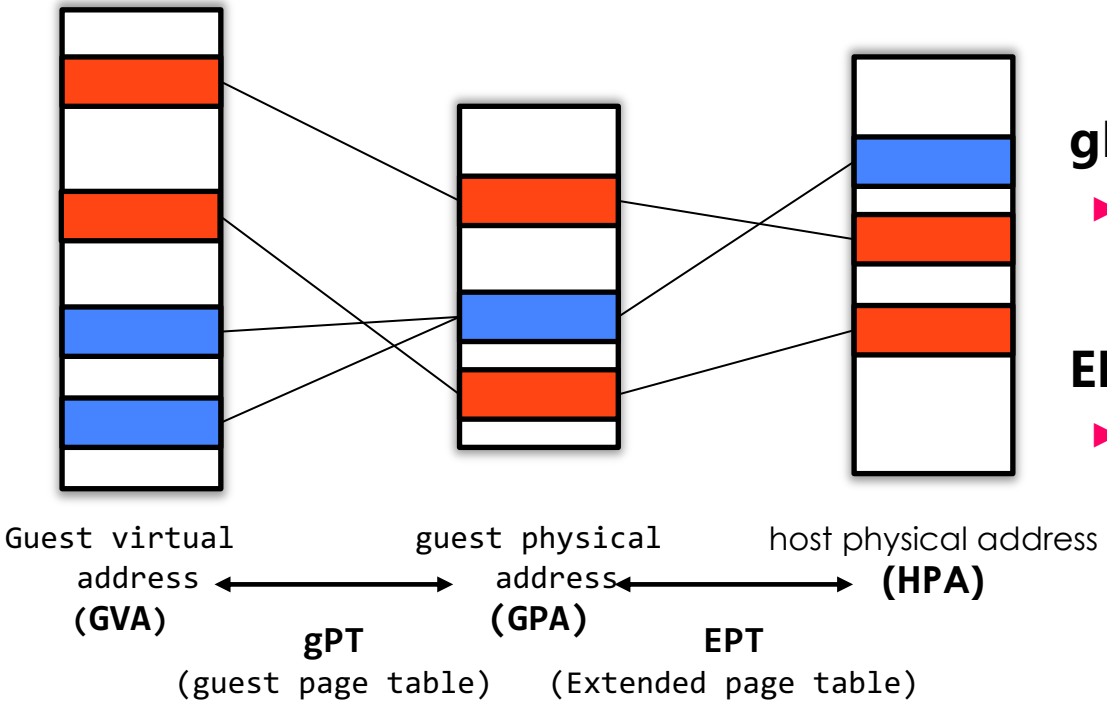
An advanced feature for hardware virtualization

1. Hardware functionality provided by Intel (with VMFUNC instruction)
2. Select an EPT from a list (configured by the hypervisor)
3. No trap into the hypervisor

Performance of VMFUNC

1. Cost 136 cycles on an Intel Skylake machine
2. Do not need to flush TLB

Two Kinds of Address Translations



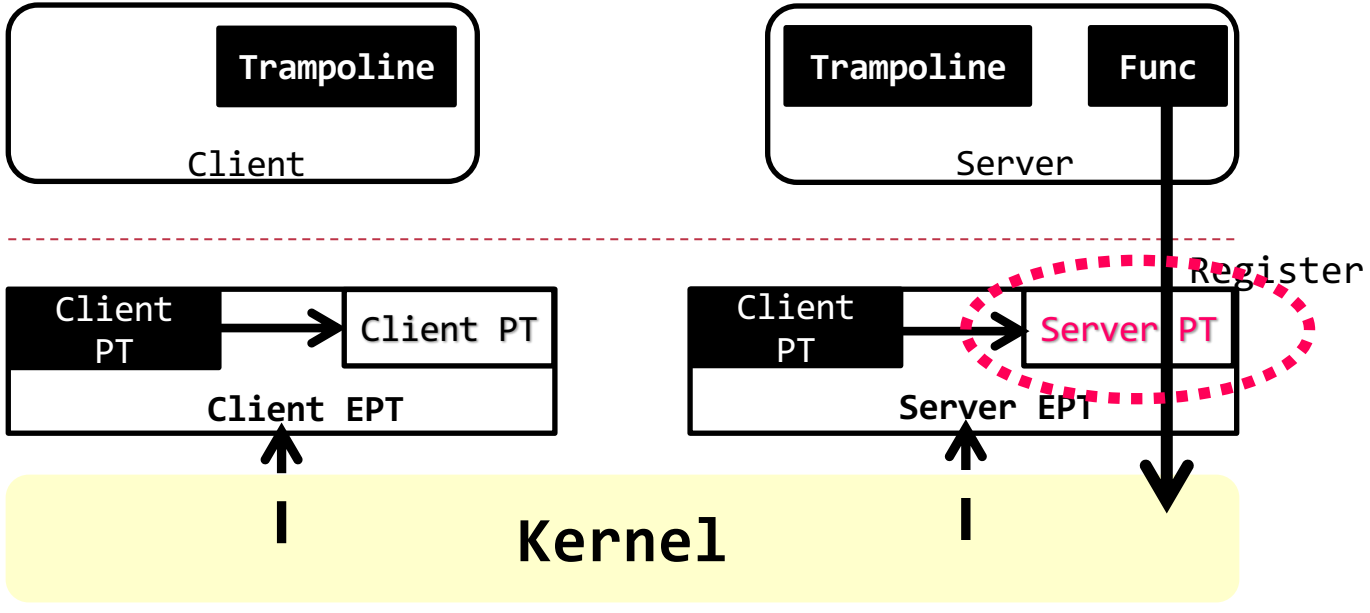
gPT translates GVA to GPA

- Controlled by the **guest**

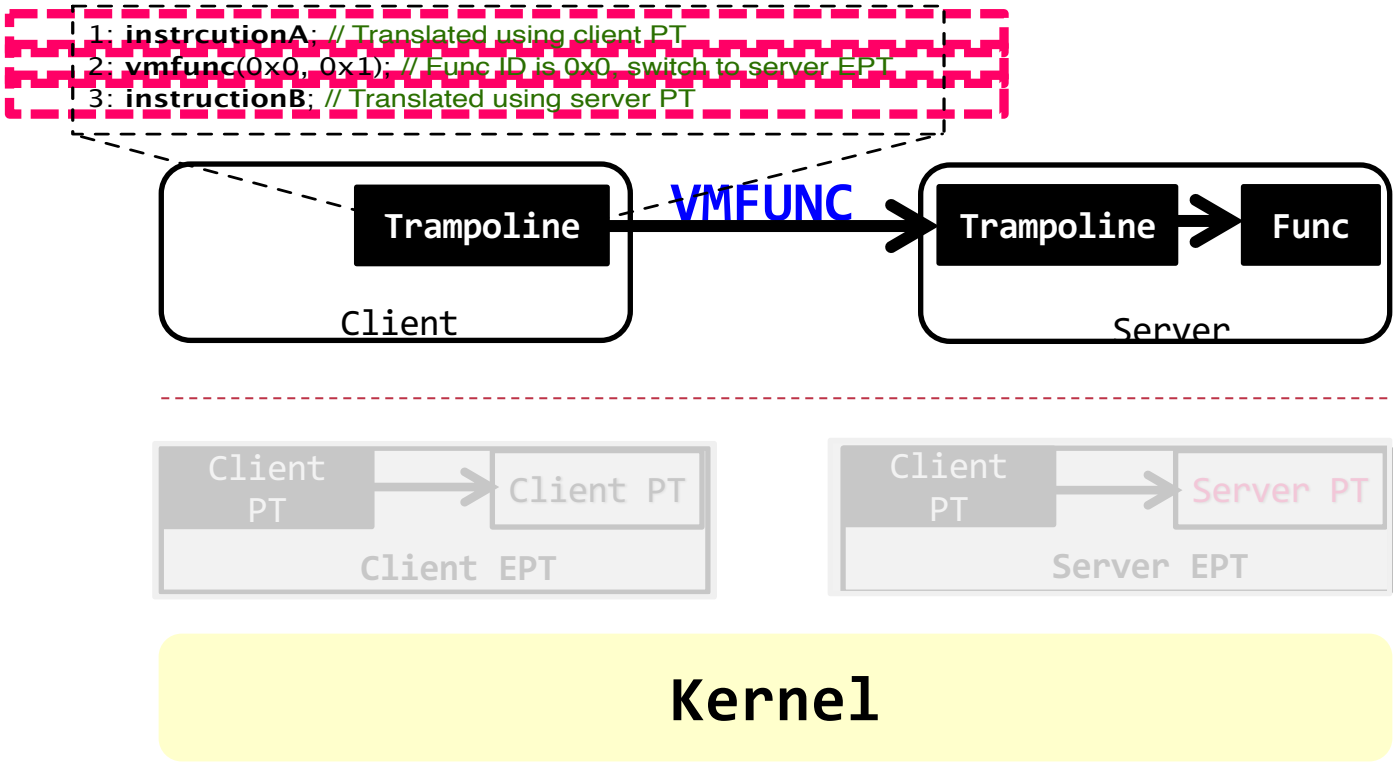
EPT translates GPA to HPA

- Controlled by the **hypervisor**

General Workflow

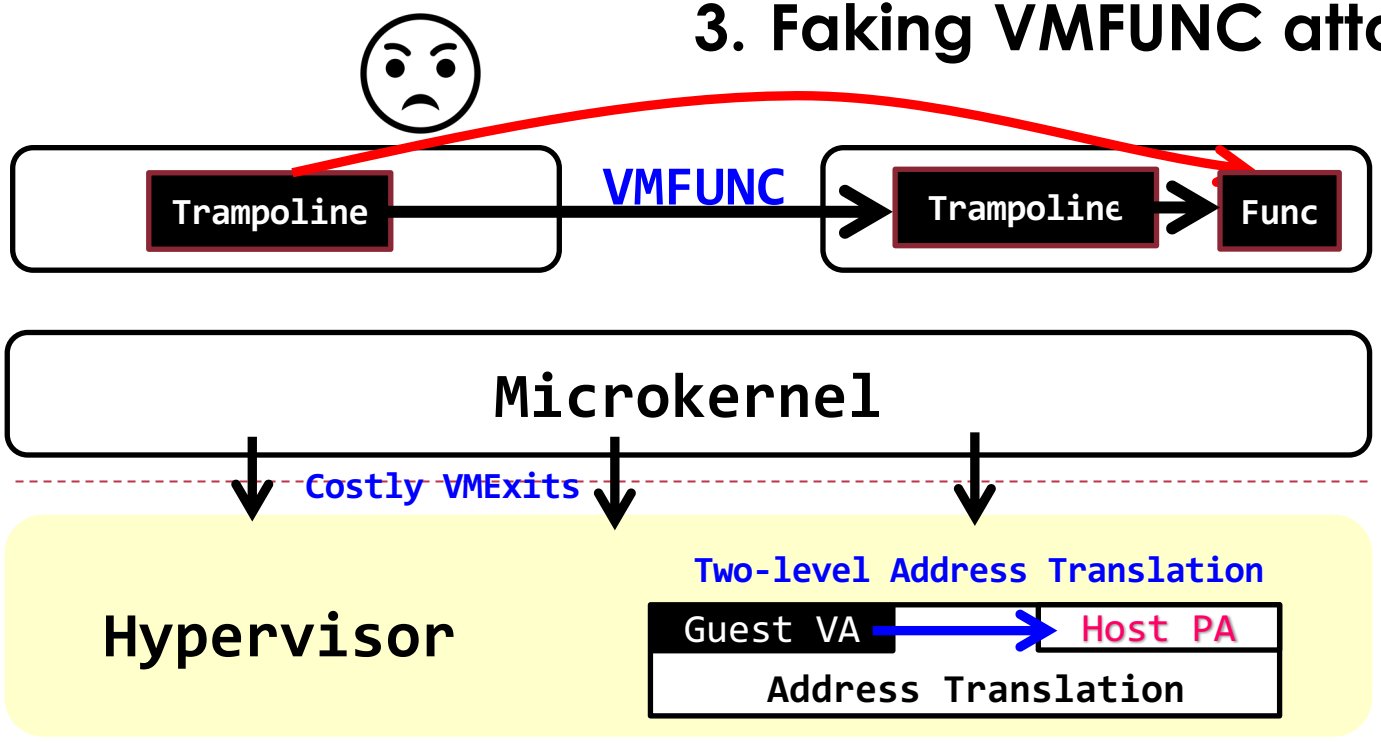


General Workflow



Challenges

1. Virtualization overhead
2. How to efficiently leverage VMFUNC
3. Faking VMFUNC attacks



Outline

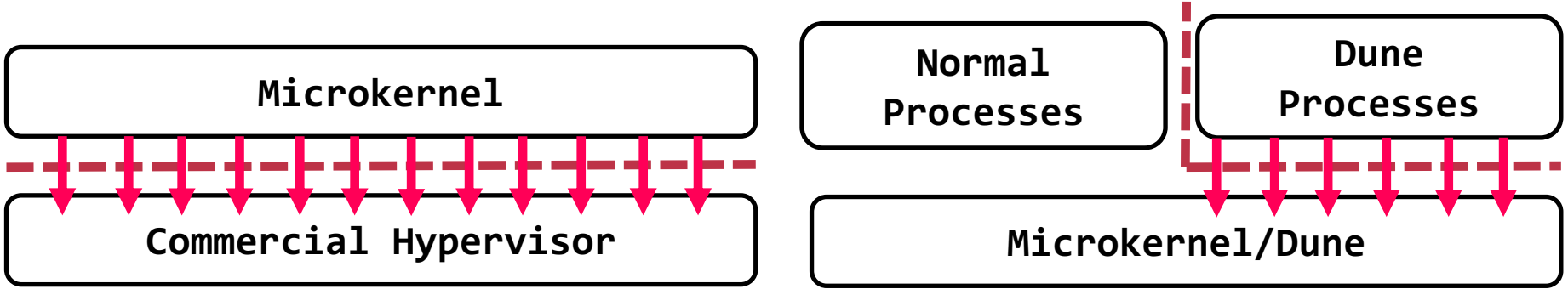
Efficient self-virtualization

Lightweight virtual address switch

Secure trampoline

Evaluation

How to Use Hardware Virtualization



Commercial hypervisor

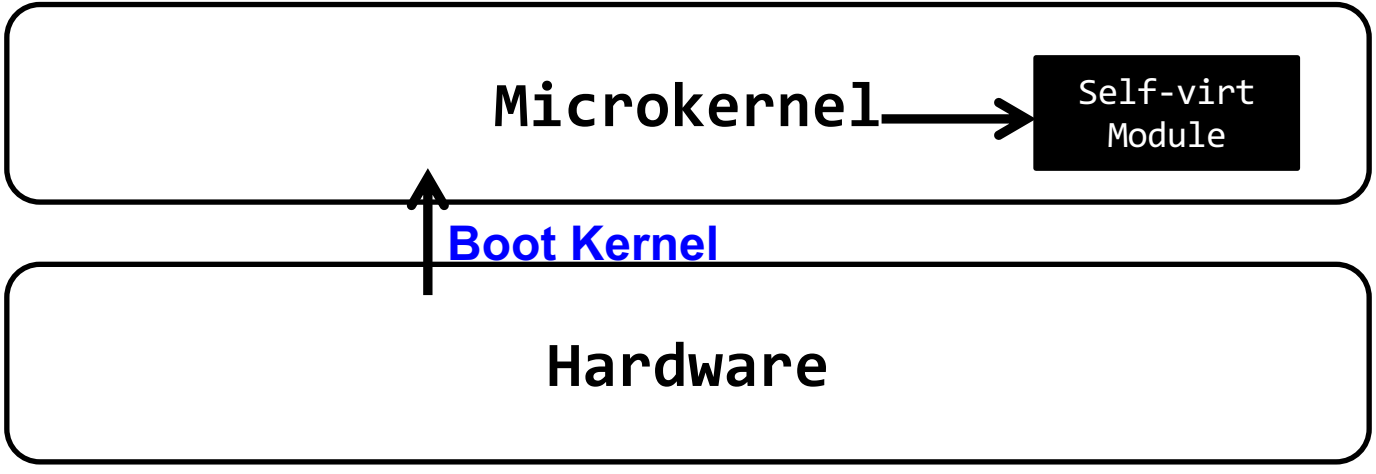
- A large number of costly VMExits
- Expensive two-level translation

Dune approach[1]

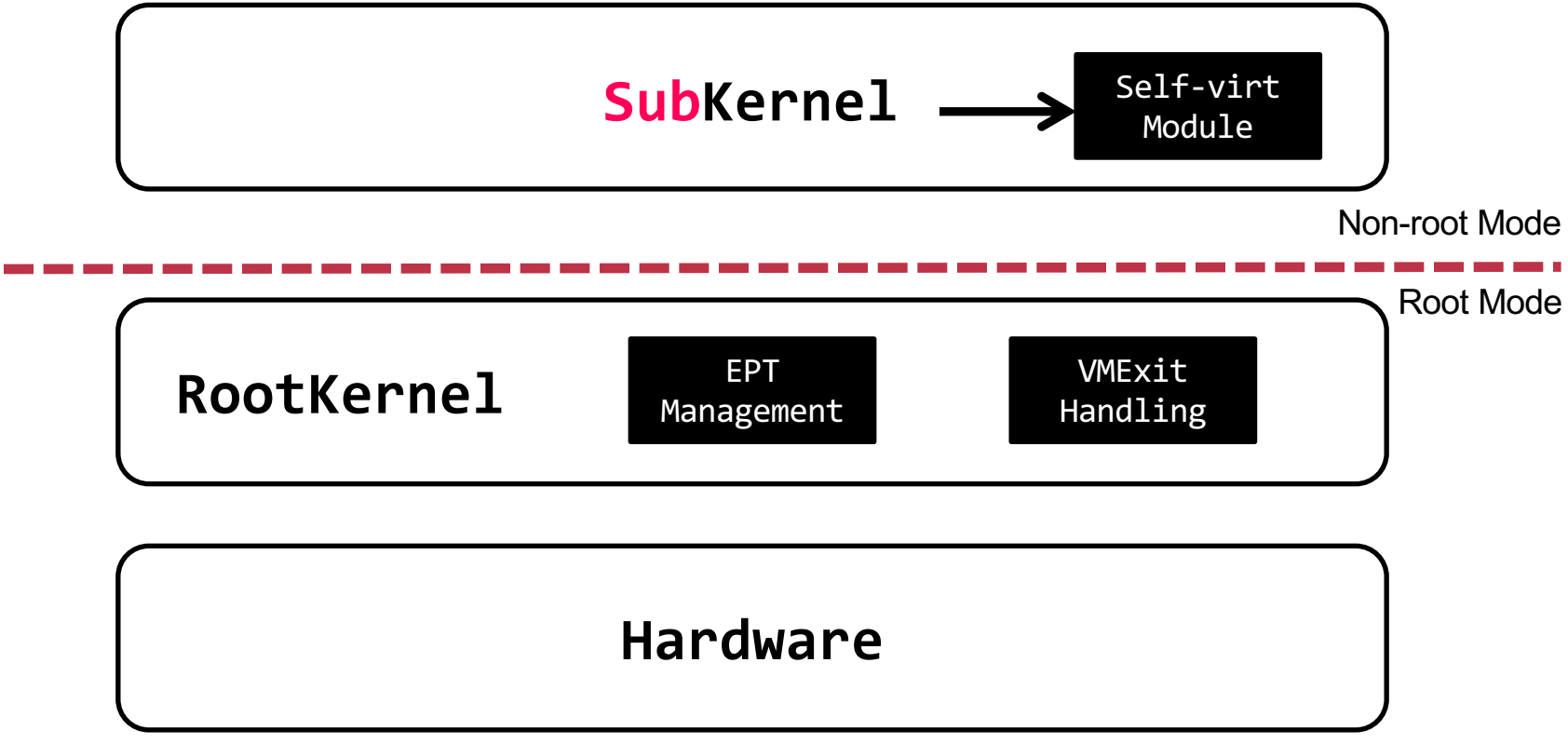
- Syscall = VMExits
- Modifying microkernels

[1] Dune: Safe User-level Access to Privileged CPU. Belay A, Bittau A, Mashtizadeh A, et al. OSDI. 2012

Efficient Self-virtualization



Efficient Self-virtualization



Efficient Self-virtualization

VMExit Handling

- Allow Subkernel to handle most hardware events (external interrupts and exceptions)
- Allow Subkernel to execute privileged instructions (e.g., HLT)
- **ZERO** VMExits in our evaluation

EPT for Subkernel

- One-to-one GPA->HPA mapping
- Using Hugepages to reduce translation and TLB missing overhead

Outline

Efficient self-virtualization

Lightweight virtual address switch

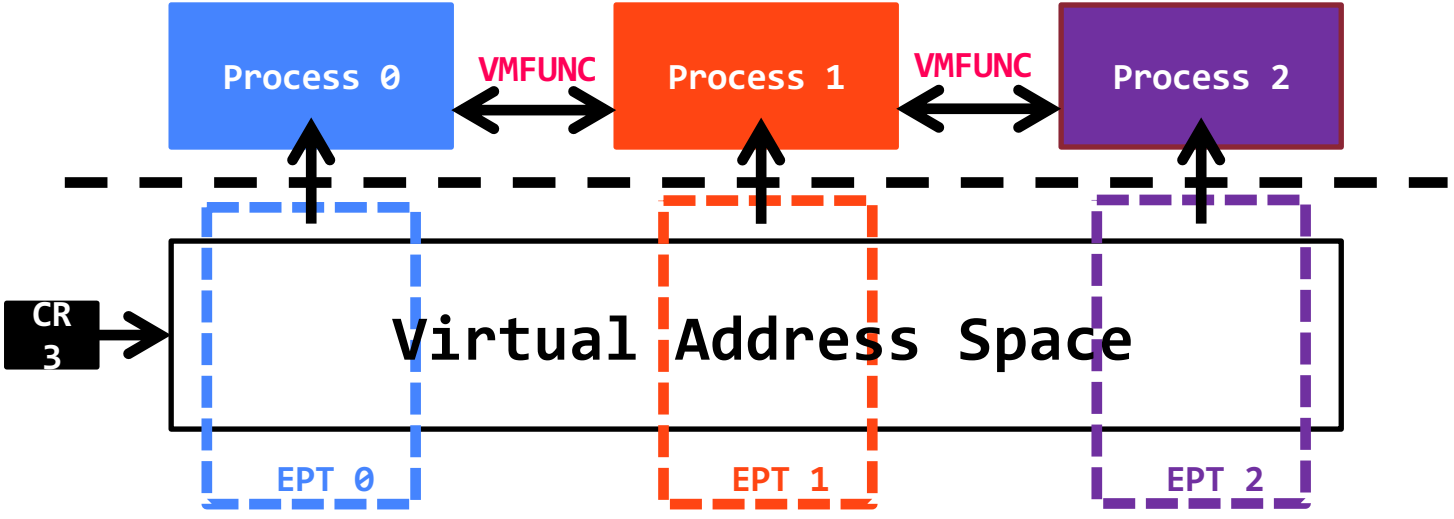
Secure trampoline

Evaluation

Traditional Approach

SeCage[1] (CCS’ 15)

- Put different processes into the same virtual address space
- Leverage different EPTs to isolate them
- Use VMFUNC to switch EPTs



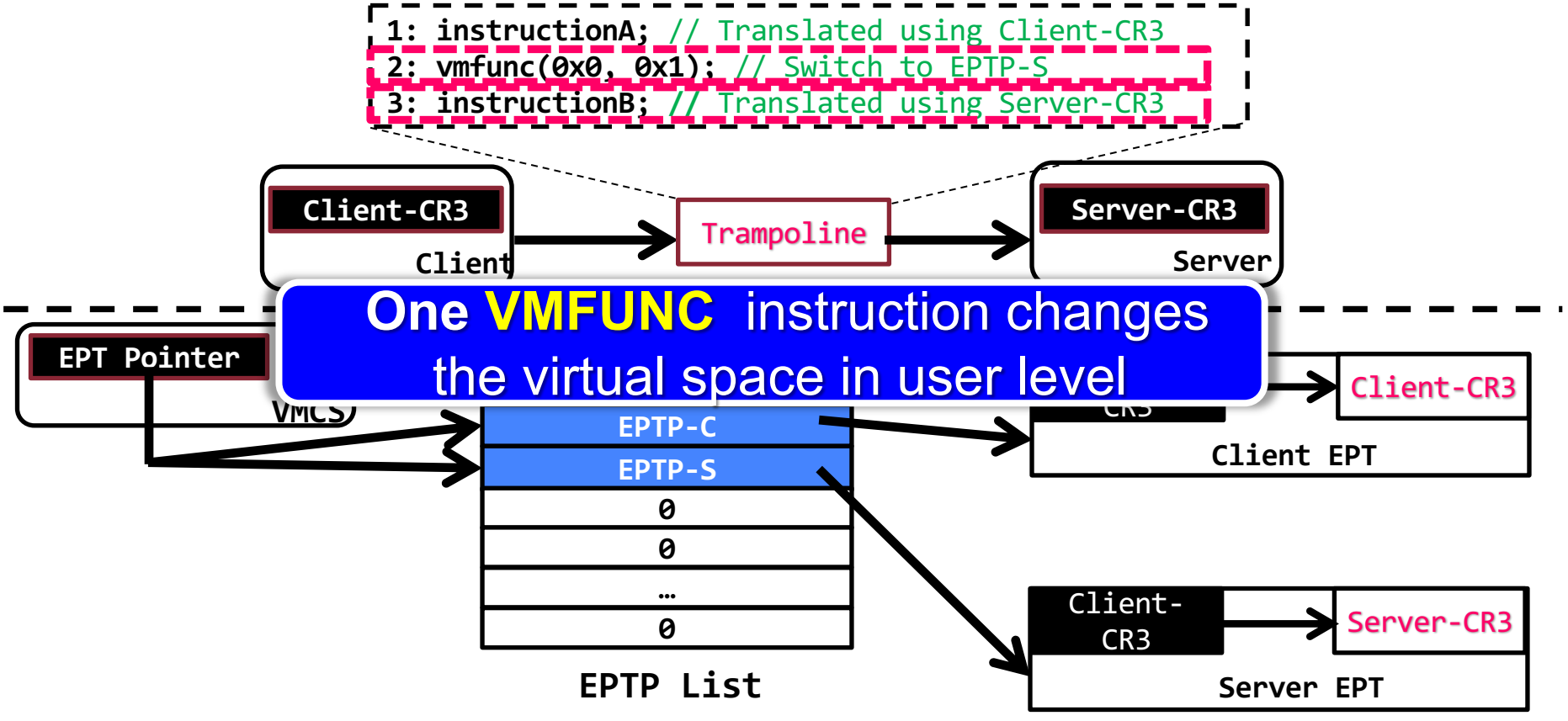
[1] Thwarting memory disclosure with efficient hypervisor-enforced intra-domain isolation. Liu Y, et al. CCS. 2015.

Traditional Approach

SeCage (CCS’ 15)

- Different processes may use the same virtual address ranges
- Carefully organize the virtual address space to prevent any possible overlap
- That requires tedious engineering efforts

Lightweight Virtual Address Switch



Outline

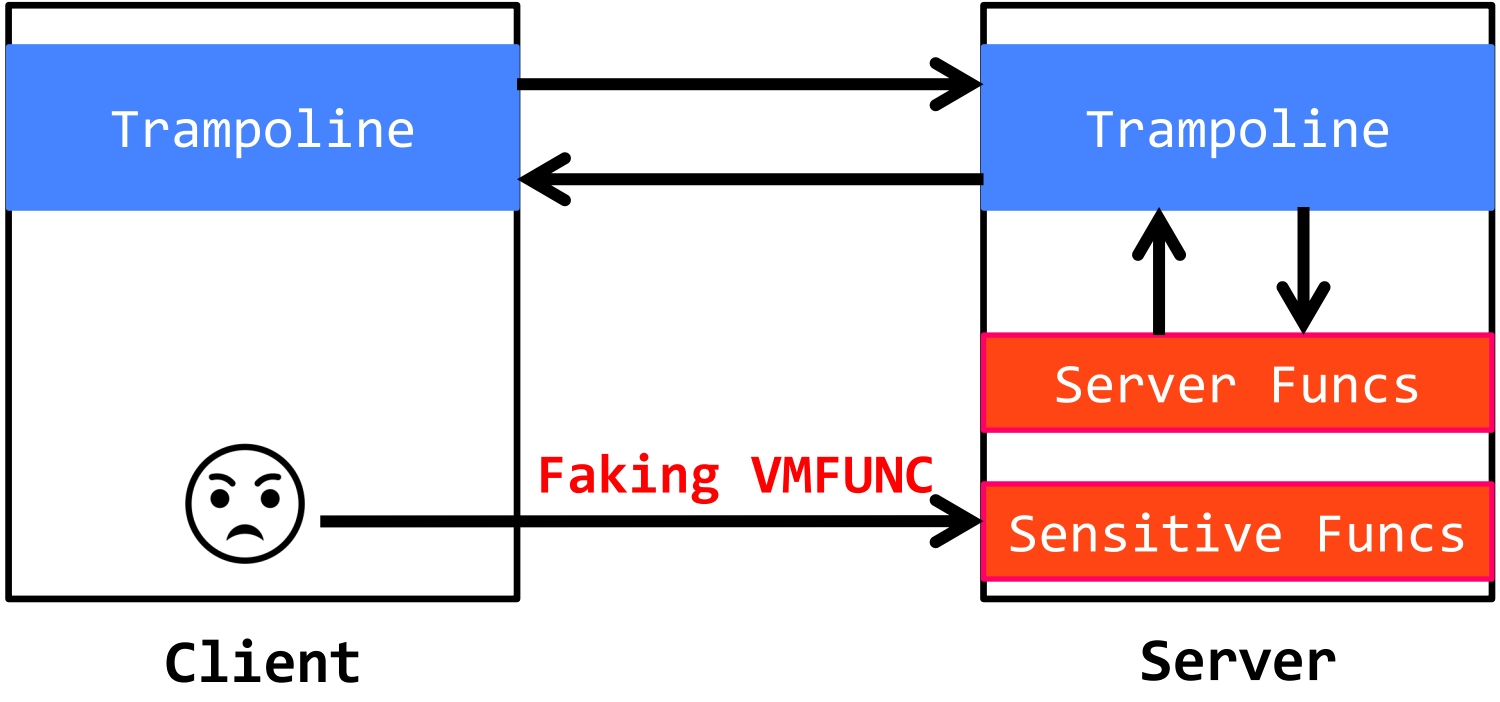
Efficient self-virtualization

Lightweight virtual address switch

Secure trampoline

Evaluation

Faking VMFUNC Attack



Secure Trampoline

- Replace all illegal VMFUNC to semantically equivalent instructions
- Replacement strategy is based on x86 variable-length instruction encoding

ID	Overlap Case	Rewriting Strategy
1	Opcode=VMFUNC	Replace VMFUNC with 3 NOP instructions
2	Mod R/M=0x0F	Push/pop used register; use new register
3	SIB=0x0F	Push/pop used register; use new register
4	Displacement=0x0F	Compute displacement value before the instruction
5	Immediate=0x0F	Apply instruction twice with different immediates to get equivalent effect
		Jump-like instruction: modify immediate after moving this instruction

Secure Trampoline

Program	Average Code Size (KB)	VMFUNC Count
SPECCPU 2006 (31 Apps)	424	0
PARSEC 3.0 (45 Apps)	842	0
Nginx v1.6.2	979	0
Apache v2.4.10	666	0
Memcached v1.4.21	121	0
Redis v2.8.17	729	0
Vmlinux v4.14.29	10,498	0
Linux Kernel Modules v4.14.29 (2,934 Modules)	15	0
Other Apps (2,605 Apps)	216	1 (in GIMP-2.8)

Outline

Efficient self-virtualization

Lightweight virtual address switch

Secure trampoline

Evaluation

Evaluation

Baseline: state-of-the-art microkernels

- seL4, Fiasco.OC, Google Zircon

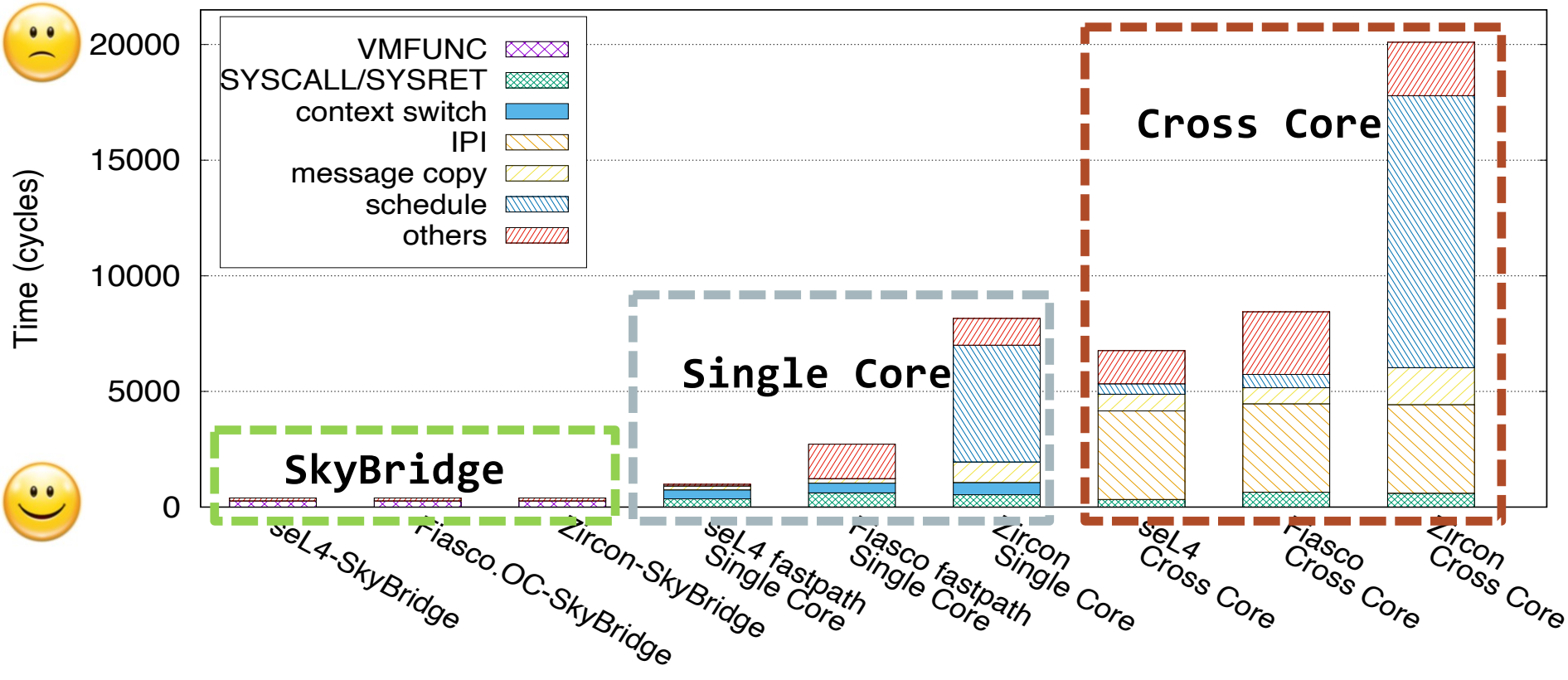
Platform:

- CPU: Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz,
  4 cores and 8 hardware threads
- Memory: 2 * 8GB DDR4

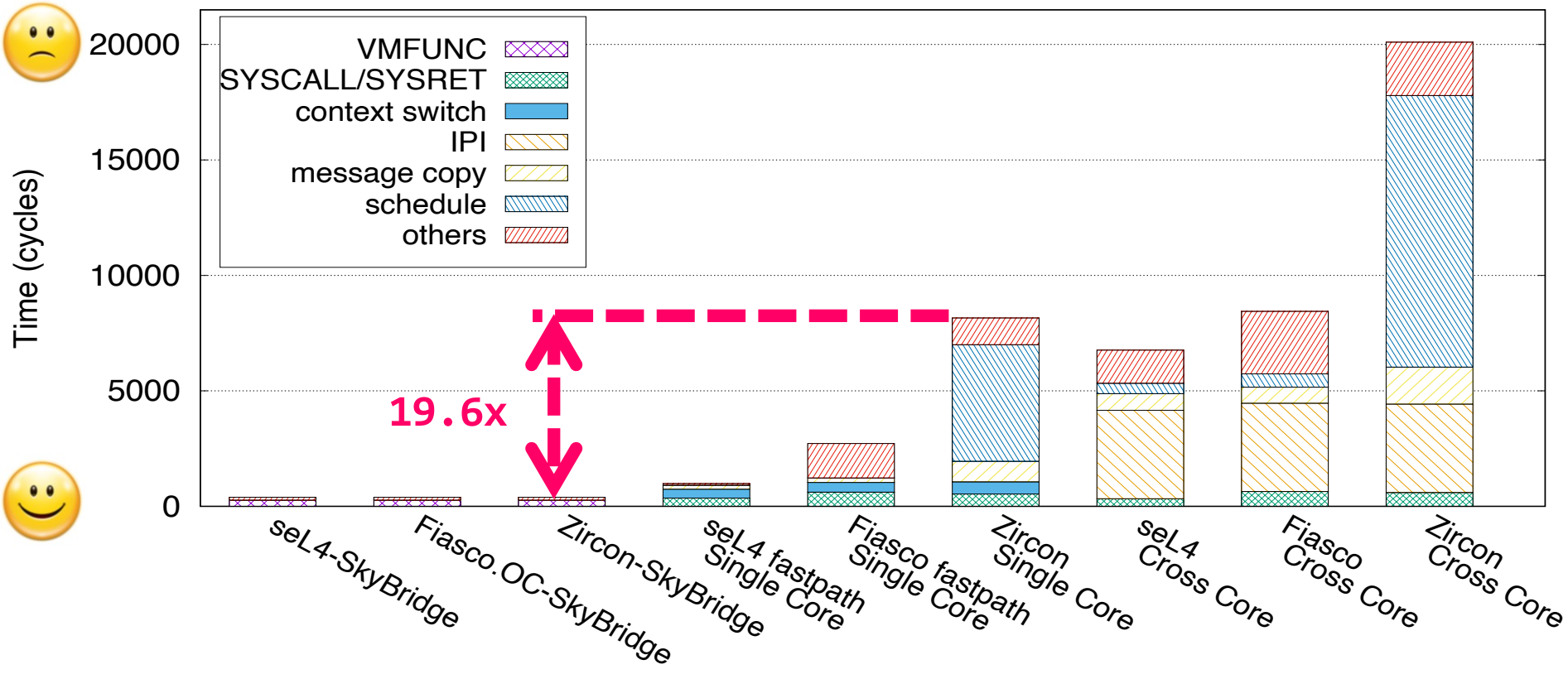
Benchmarks

- Synthetic: KVS
- Real-world: SQLite 3.0

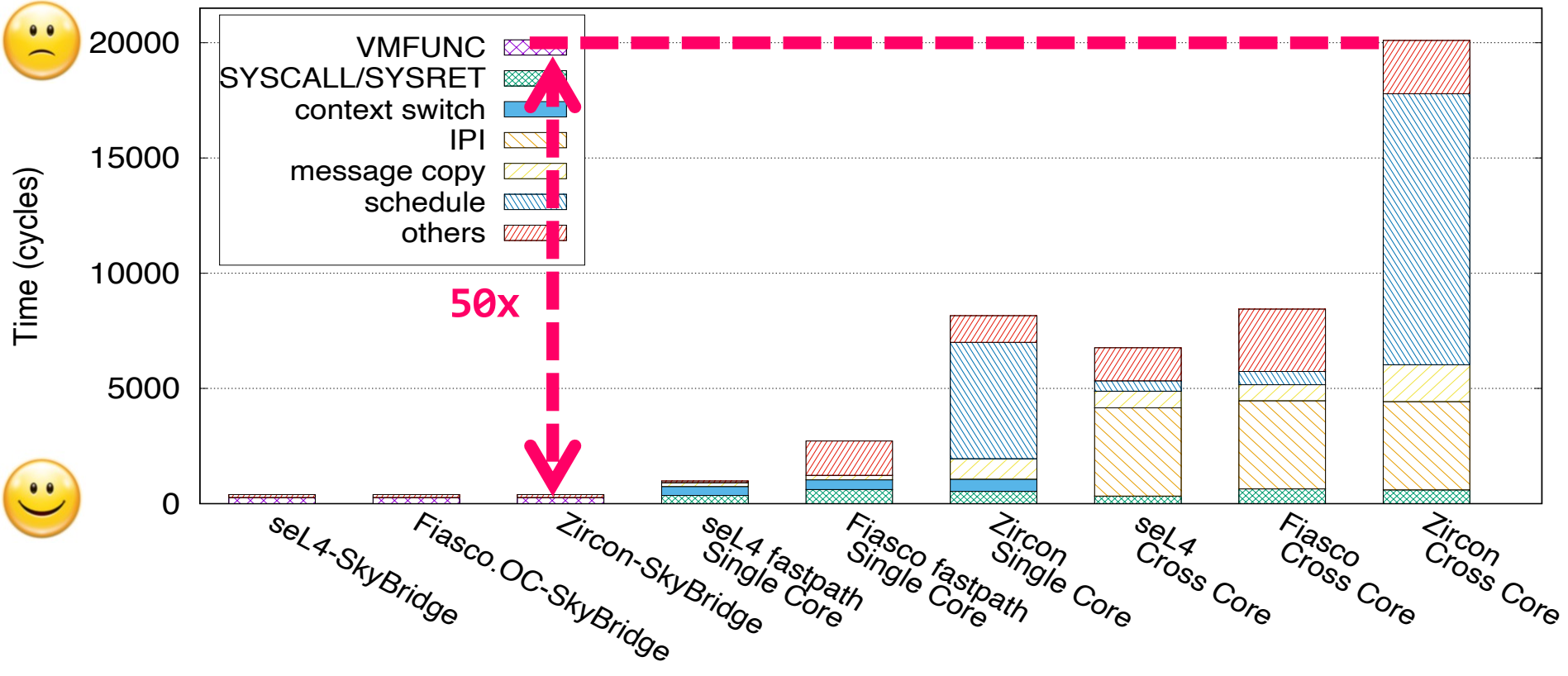
IPC Performance



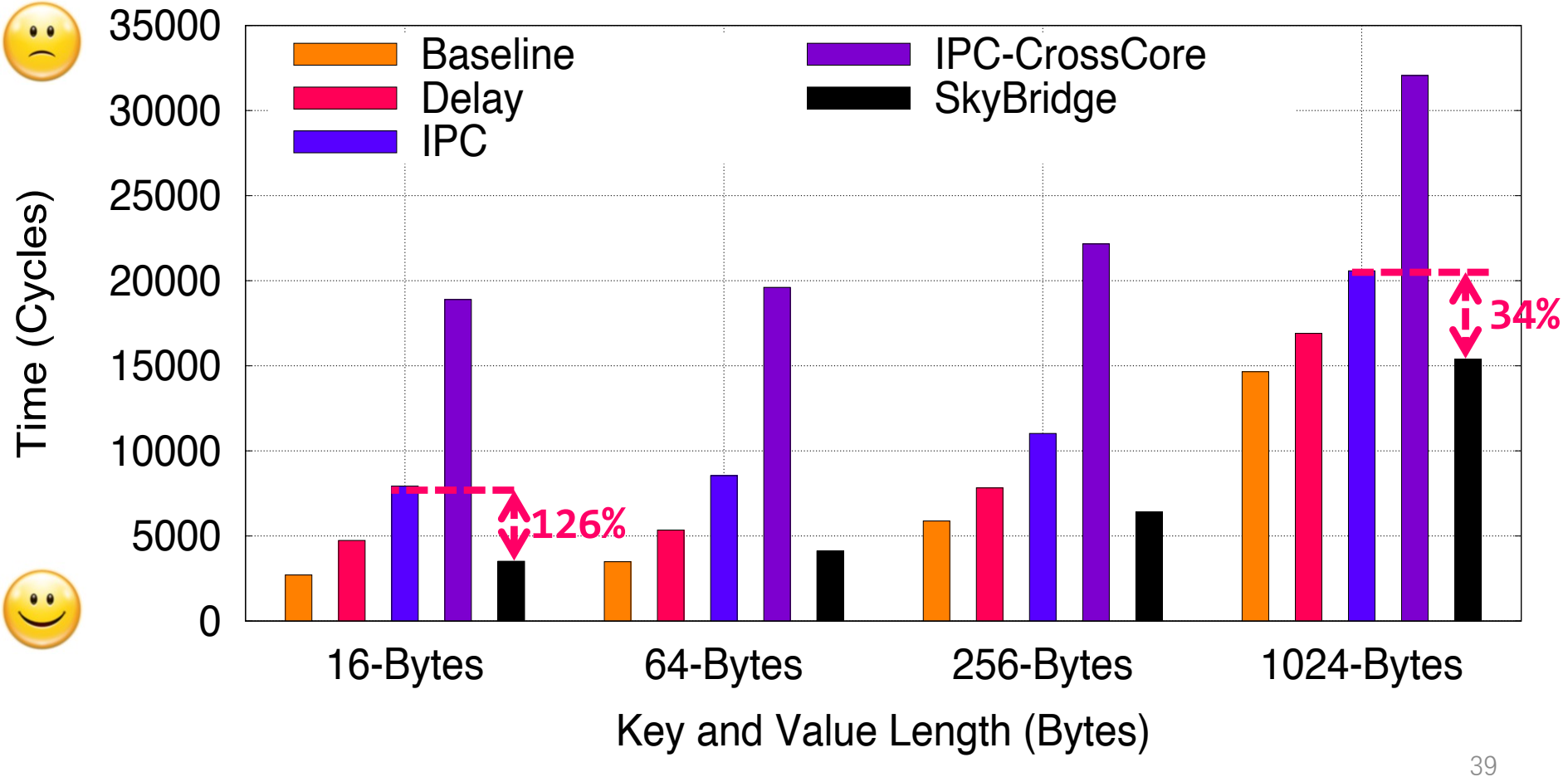
IPC Performance



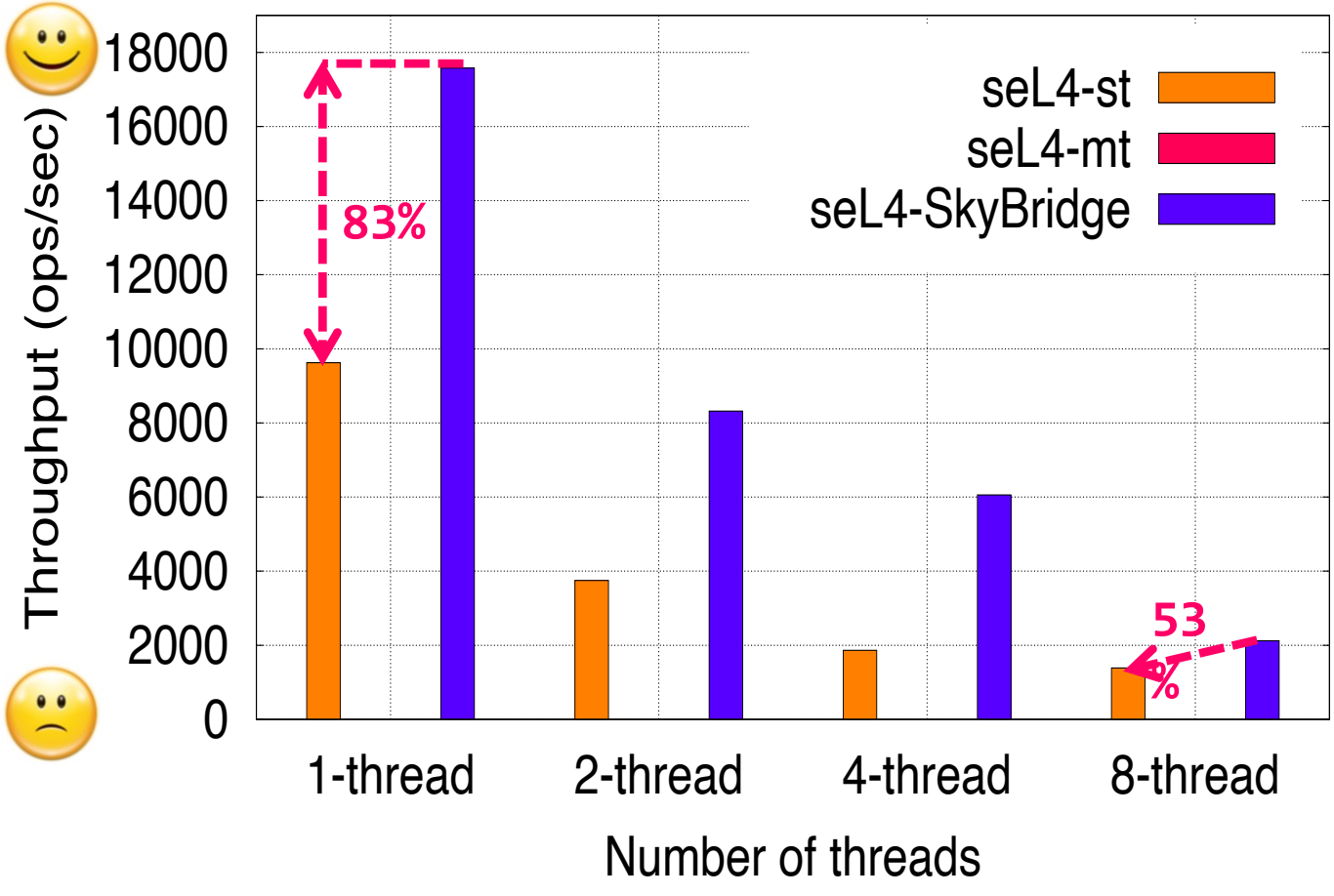
IPC Performance



Performance of KVS

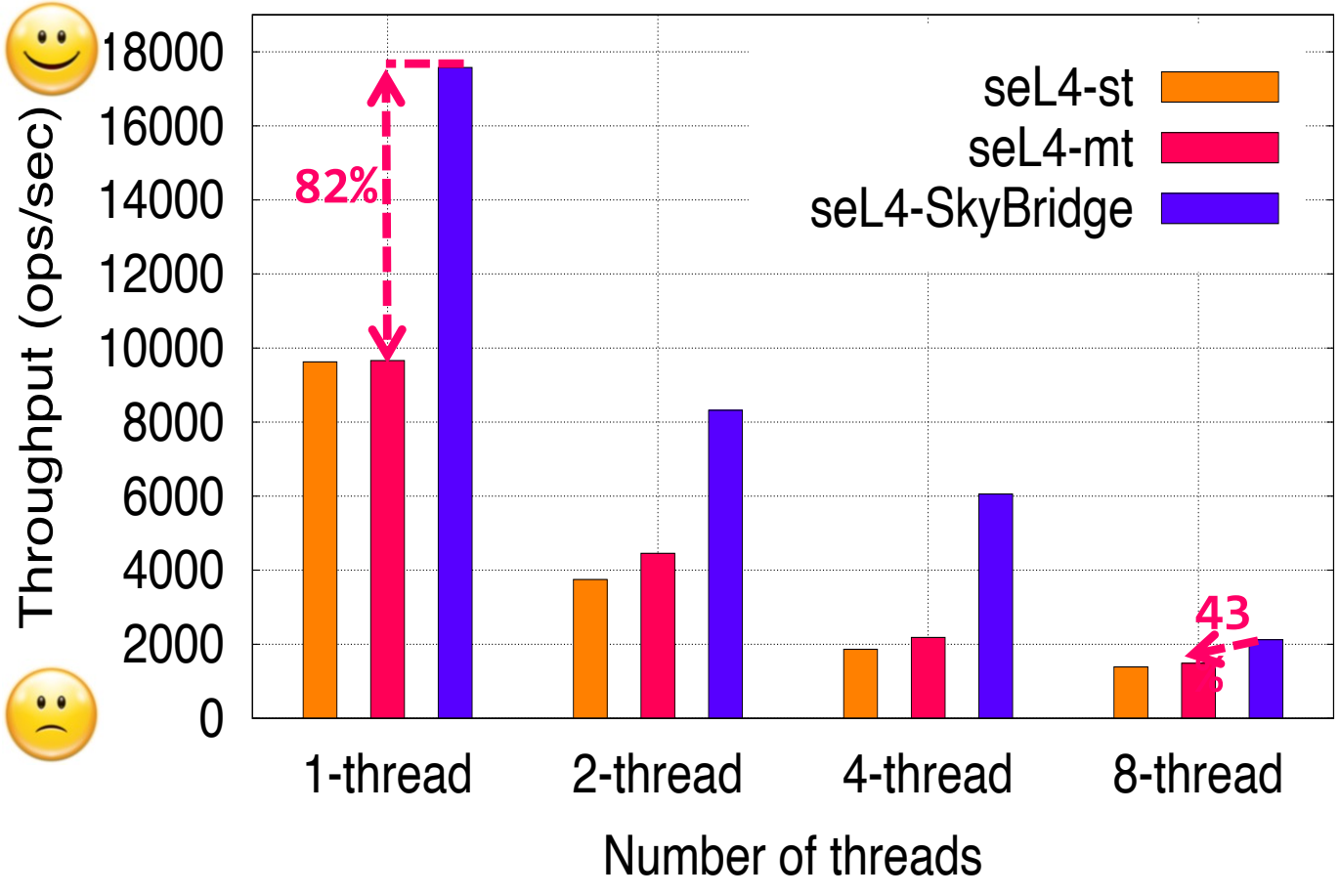


Performance of SQLite 3.0



seL4-st:
Multiple SQLite threads
One server thread

Performance of SQLite 3.0



seL4-st:
Multiple SQLite threads
One server thread

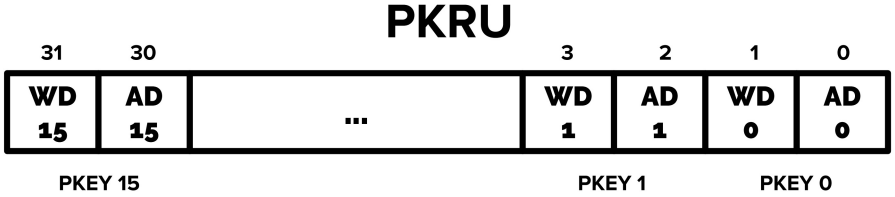
seL4-mt:
Multiple SQLite threads Multiple server threads

Harmonizing Performance and Isolation in Microkernels with Efficient Intra-kernel Isolation and Communication (USENIX ATC 2021)

PKU Brings New Opportunities

- **PKU: Protection Key for Userspace (aka. MPK)**
  - Assign each memory page one PKEY (i.e., domain ID)

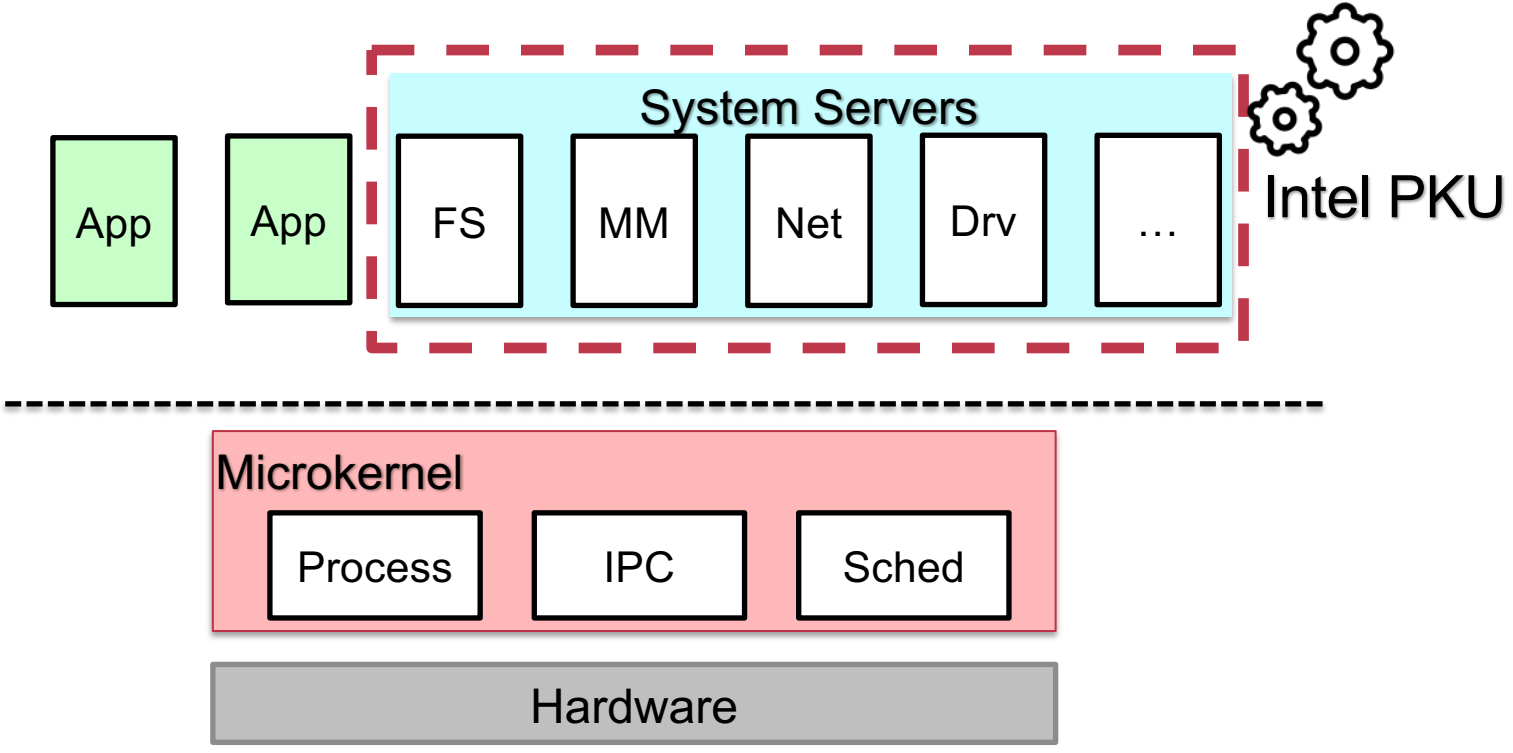
  - A new register PKRU stores read/write permission



Efficient Intra-Process Isolation

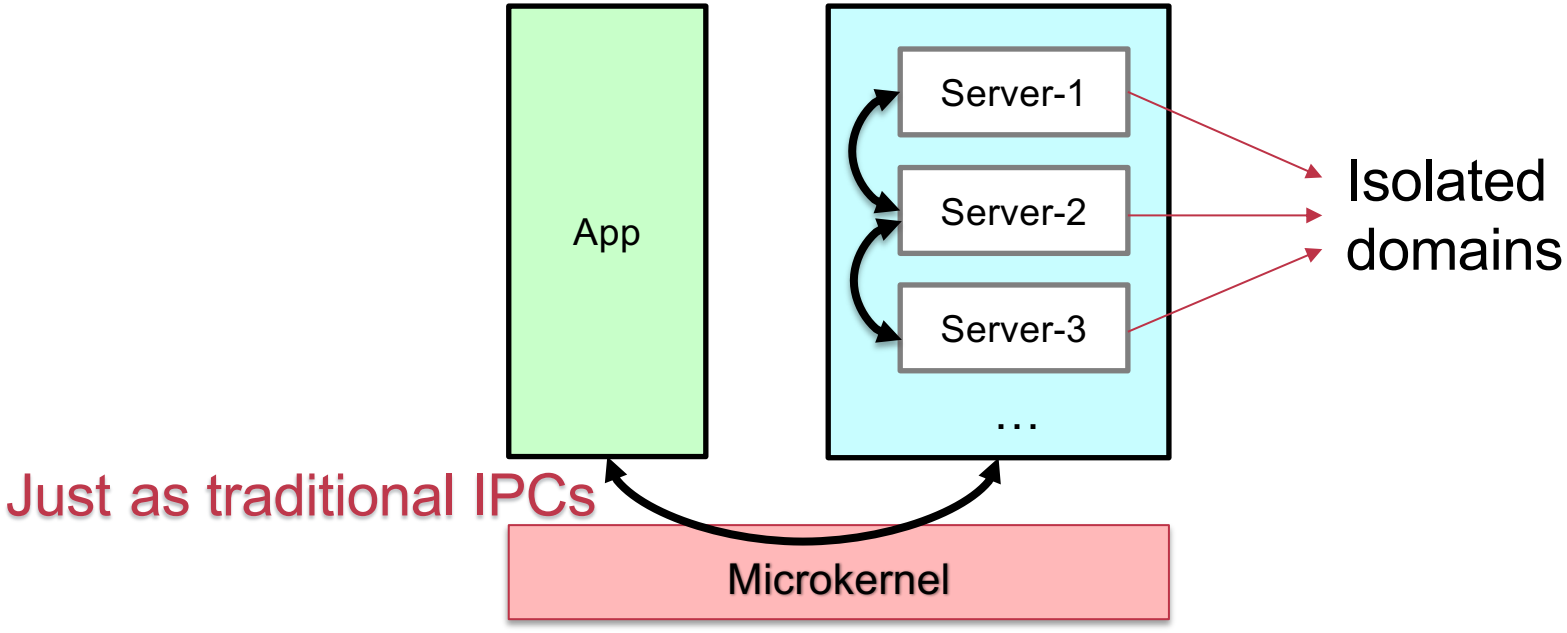
- ERIM [Security' 19] & Hodor [ATC' 19] (Intel PKU)
  - **More efficient**
  - **Same address space**
  - Domain switch only takes **28 cycles**

Intra-Process Isolation + Microkernel



Design Choice #1

Isolate different system servers in a single process.

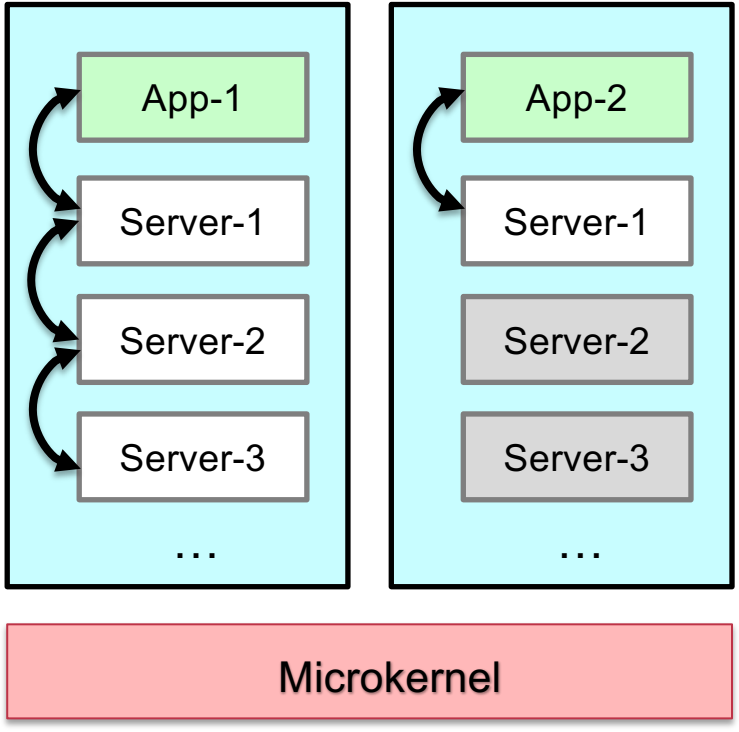


Design Choice #2

Let's get more aggressive!

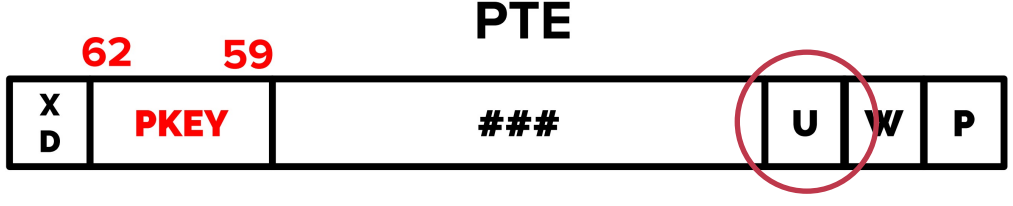
Drawbacks

1. Update Server mapping is costly
2. IPC connection is also costly
3. Less flexibility for applications on address space and using PKU

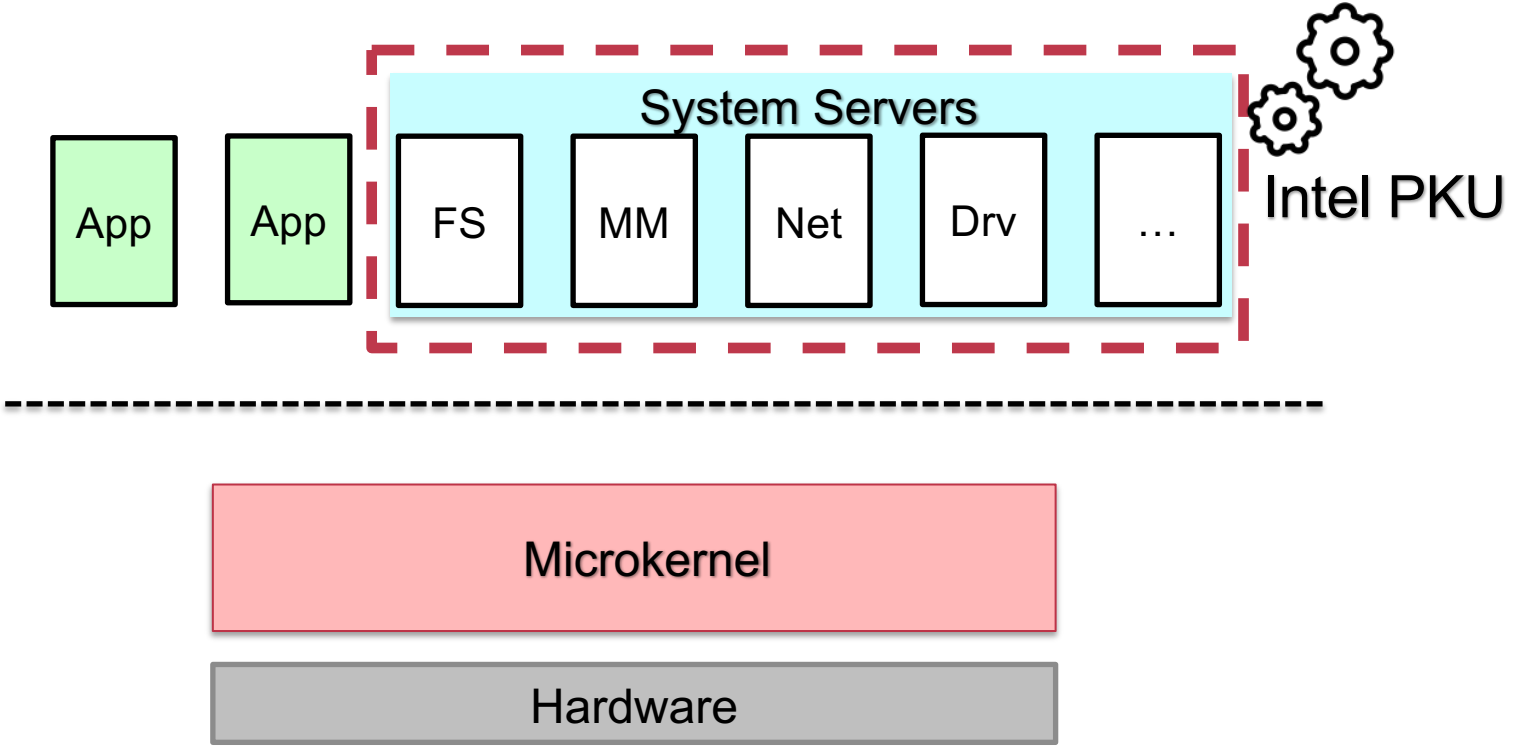


An Observation on Intel PKU

- **A misleading name**
  - Protection Key for Userspace
- **It still takes effect when in kernel (ring-0)**
  - The "Userspace" means user-accessible memory
  - U/K bit in PTE

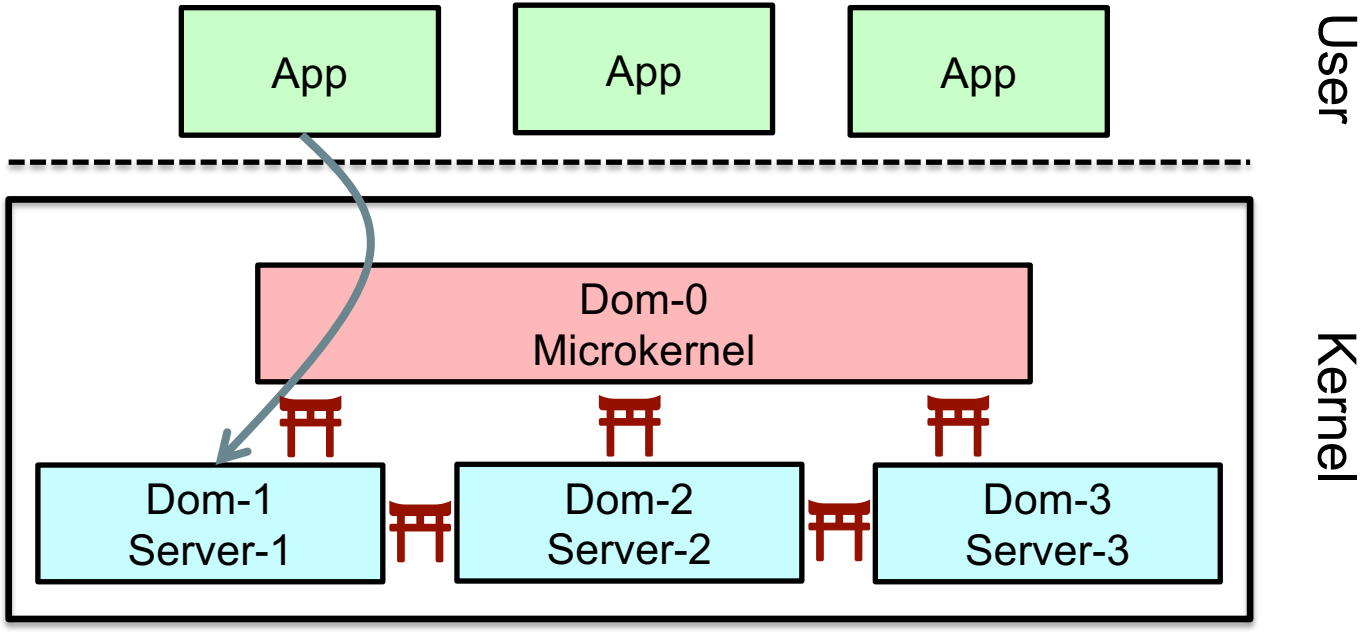


UnderBridge: Sinking System Servers



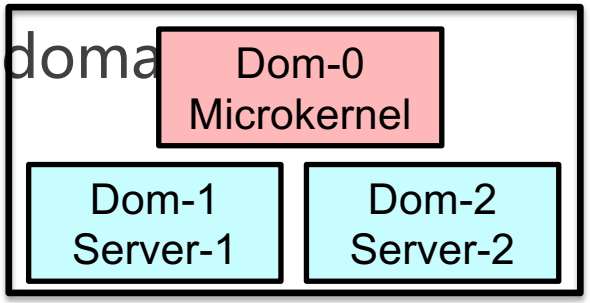
Design Choice #3: UnderBridge

- Build **execution domains** in the **kernel page table**



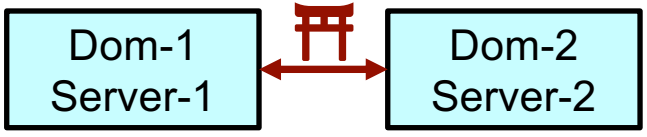
Execution Domain

- **Execution domain 0 is for the microkernel**
  - Use memory domain 0
  - Can access all the memory
- **Others own a private memory domain**
  - A private MPK memory domain ID
- **Shared memory**
  - Allocate a free MPK memory doma

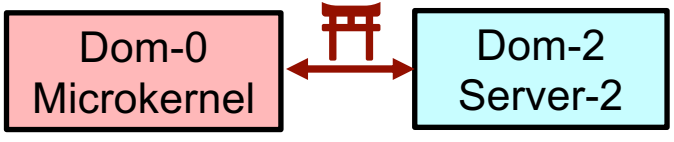


IPC Gate

- **Connect two servers**
  - Generated by the microkernel
  - Resides in memory domain 0 (execute-only for servers)

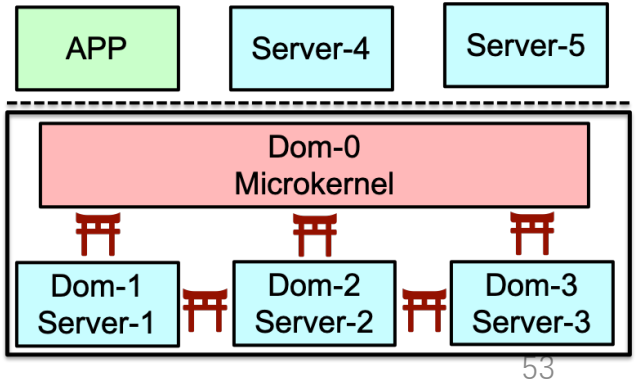


- **Transfer control flow during IPC invocations**
  - context switch and domain switch
- **Connect the microkernel and servers**
  - System calls -> IPC gates



Server Migration

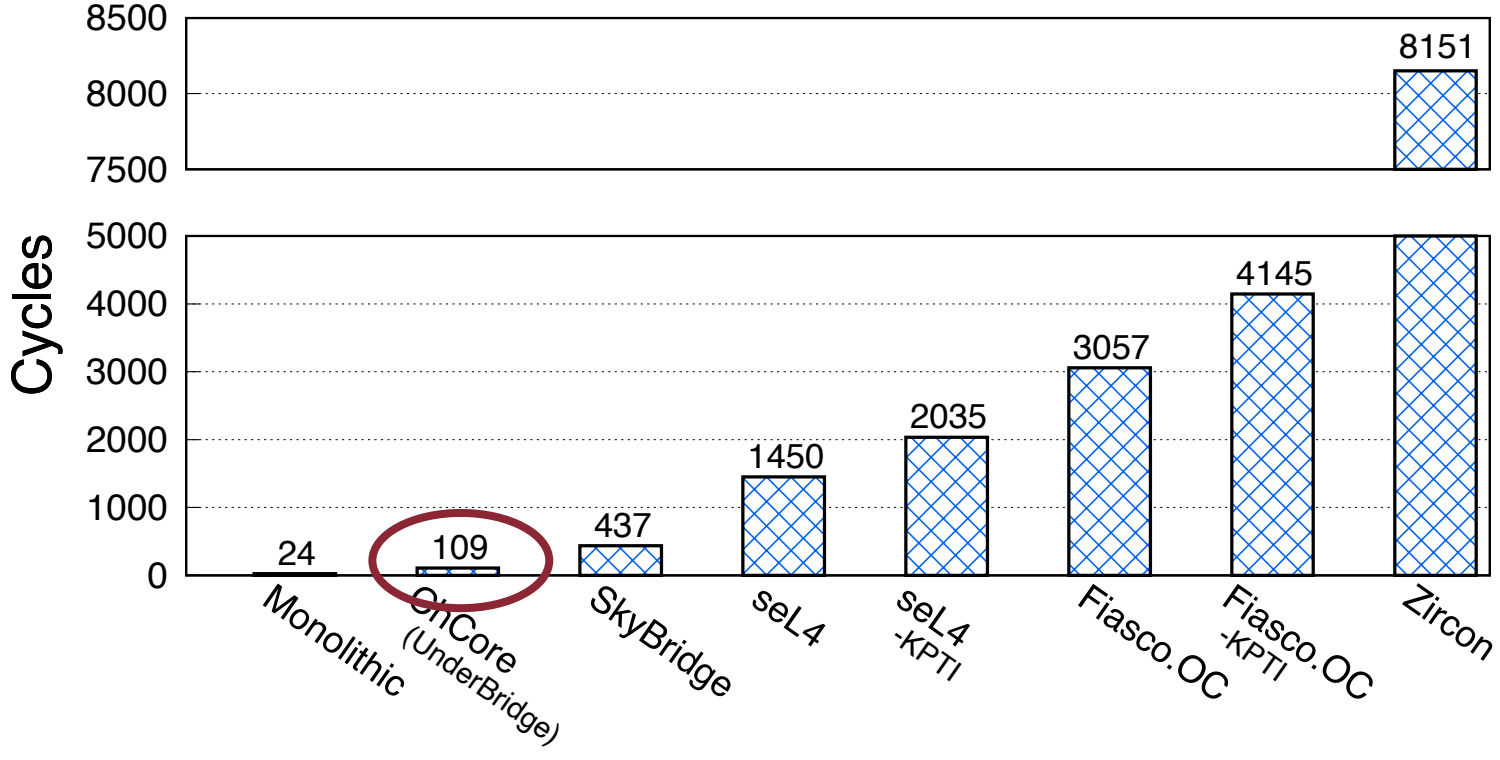
- **The number of execution domain is limited**
  - Hardware only provides 16 memory domains
  - Time-multiplexing is expensive
- **Move servers between user and kernel space**
  - Disjoint virtual memory regions
  - Runtime migration



Privilege Deprivation

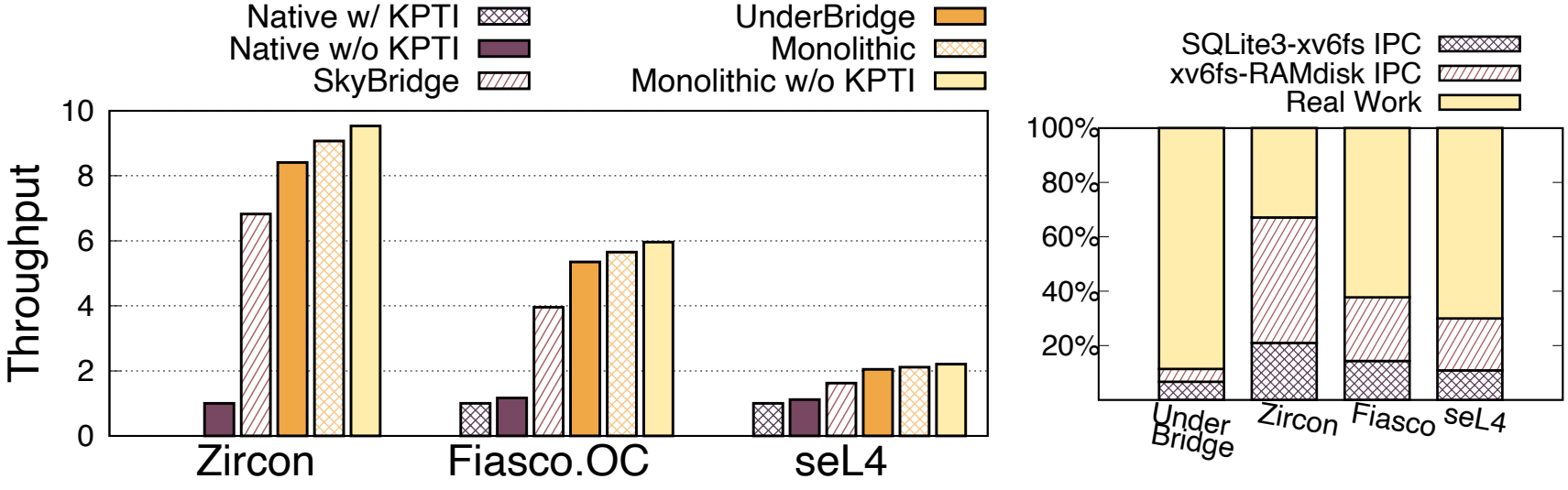
- **In-kernel servers have supervisor privilege**
  - Can affect the whole system if compromised
  - CFI (with binary scanning) incurs runtime overhead
  - Binary rewriting only is infeasible
- **Prevent servers to execute privilege instructions**
  - Add a tiny secure monitor in hypervisor mode
  - For instructions rarely execute: VMExits
  - For instructions that frequently required: Rewriting

Cross-server IPC Round-Trip Latency



Evaluated on Dell PowerEdge R640 server with Intel Xeon Gold 6138 CPU

SQLite Throughput under YCSB-A



Conclusion

- IPC is the Achille's heel of microkernels.
- SkyBridge
  - Bypassing the OS kernel by leveraging VMFUNC
  - Strong isolation via EPT domains
- UnderBridge
  - Putting deprivileged servers back to kernel and isolate them with Intel PKU
  - Faster domain switch

SCAN ME

Thanks!

IPADS INSTITUTE OF PARALLEL AND DISTRIBUTED SYSTEMS