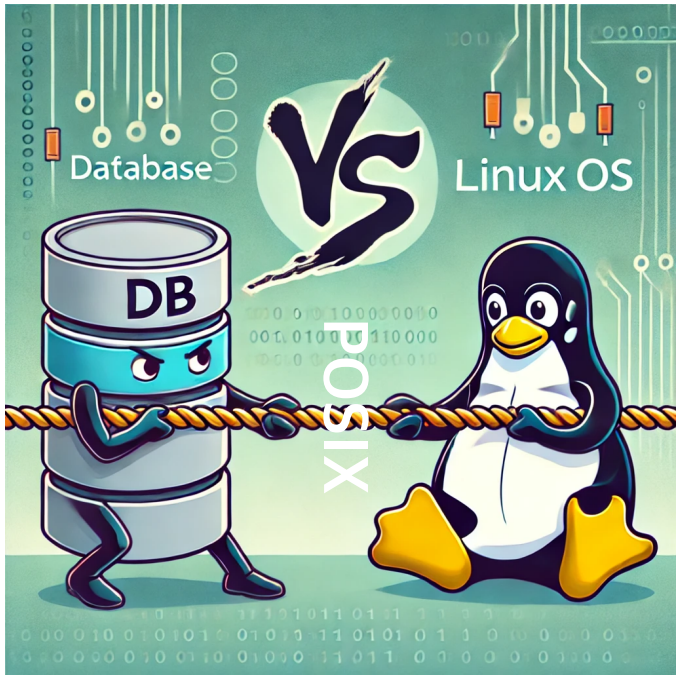
Practical DB-OS Co-Design with Privileged Kernel-Bypass

Xinjing Zhou, Viktor Leis, Jinming Hu, Xiangyao Yu, Michael Stonebraker

MIT CSAIL, TUM, DolphinDB Inc, UW-Madison, MIT CSAIL

DB-OS Interface Mismatch

Performance
Hardware control



Security
Multiplexes hardware
Resource efficiency

DBMS Reimplements a lot of OS Features

DBMS Reimplements a lot of OS Features

- File/Storage caching

DBMS Reimplements a lot of OS Features

- File/Storage caching
- Snapshotting

DBMS Reimplements a lot of OS Features

- File/Storage caching
- Snapshotting
- User-space task scheduling and I/O scheduling

DBMS Reimplements a lot of OS Features

- File/Storage caching
- Snapshotting
- User-space task scheduling and I/O scheduling
- But DBMS, being unprivileged, does not have powers that OS has
  - Page Table, MMU
  - Hardware interrupts
  - TLB flush instructions
  - Only exposed through slow POSIX interfaces (mmap, madvise, signals)

Notes on Data Base Operating Systems

Author:  Jim Gray Authors Info & Claims

Operating Systems, An Advanced Course • January 1978 • Pages 393–481

Published: 01 January 1978 Publication History

Operating system support for database management

Author:  Michael Stonebraker Authors Info & Claims

Communications of the ACM, Volume 24, Issue 7 • pp 412–418 • https://doi.org/10.1145/358699.358703

Notes on Data Base Operating Systems

Author:  Jim Gray Authors Info & Claims

Operating Systems, An Advanced Course • January 1978 • Pages 393–481

Published: 01 January 1978 Publication History

COD: Database / Operating System Co-Design

Jana Giceva, Tudor-Ioan Salomie, Adrian Schüpbach
Gustavo Alonso, Timothy Roscoe
Systems Group, Department of Computer Science
ETH Zurich, Switzerland
www.systems.ethz.ch

Operating system support for database r

Author: Michael Stonebraker Authors Info & Claims

Communications of the ACM, Volume 24, Issue 7 • pp 412–418 • https://doi.org/10.1145/358699.358703

Notes on Data Base Operating Systems

Author: Jim Gray Authors Info & Claims

Operating Systems, An Advanced Course • January 1978 • Pages 393–481

Published: 01 January 1978 Publication History

579 0

Async-fork: Mitigating Query Latency Spikes Incurred by the Fork-based Snapshot Mechanism from the OS Level

Pu Pang
Shanghai Jiao Tong University
Alibaba Group
avengerispp@sjtu.edu.cn

Gang De
Alibaba Gr
denggang.dg@alib

COD: Database / Operating System Co-Design

Jana Giceva, Tudor-Ioan Salomie, Adrian Schüpbach
Gustavo Alonso, Timothy Roscoe
Systems Group, Department of Computer Science
ETH Zurich, Switzerland
www.systems.ethz.ch

Operating system support for database r

Author:  Michael Stonebraker Authors Info & Claims

Communications of the ACM, Volume 24, Issue 7 • pp 412–418 • https://doi.org/10.1145/358699.358703

Notes on Data Base Operating Systems

Author:  Jim Gray Authors Info & Claims

Operating Systems, An Advanced Course • January 1978 • Pages 393–481

Published: 01 January 1978 Publication History

Async-fork: Mitigating Query Latency Spikes Incurred by the Fork-based Snapshot Mechanism from the OS Level

Pu Pang
Shanghai Jiao Tong University
Alibaba Group
avengerispp@sjtu.edu.cn

Gang De
Alibaba Gr
denggang.dg@alib

Operating system support for database r

Author: Michael Stonebraker Authors Info & Claims

Communications of the ACM, Volume 24, Issue 7 • pp 412–418 • https://doi.org/10.1145/358699.358703

COD: Database / Operating System Co-Design

Jana Giceva, Tudor-Ioan Salomie, Adrian Schüpbach
Gustavo Alonso, Timothy Roscoe
Systems Group, Department of Computer Science
ETH Zurich, Switzerland
www.systems.ethz.ch

Notes on Data Base Operating Systems

Author: Jim Gray A

Operating Systems, An Ac

Published: 01 January 19

579 0

Are You Sure You Want to Use MMAP in Your Database Management System?

Andrew Crotty
Carnegie Mellon University

Viktor Leis
Friedrich-Alexander-Universität

Andy Pavlo
Carnegie Mellon University

Async-fork: Mitigating Query Latency Spikes Incurred by the Fork-based Snapshot Mechanism from the OS Level

Pu Pang
Shanghai Jiao Tong University
Alibaba Group
avengerispp@sjtu.edu.cn

Gang De
Alibaba Gr
denggang.dg@alib

COD: Database / Operating System Co-Design

Jana Giceva, Tudor-Ioan Salomie, Adrian Schüpbach

Operating system support for

Author: Michael Stonebraker Authors Info & Claims

Communications of the ACM, Volume 24, Issue 7 • pp 412–418

Virtual-Memory Assisted Buffer Management

Preprint accepted for publication at SIGMOD 2023

Viktor Leis
Technische Universität München
leis@in.tum.de

Adnan Alhomssi
Friedrich-Alexander-Universität
Erlangen-Nürnberg
adnan.alhomssi@fau.de

Tobias Ziegler
Technische Universität Darmstadt
tobias.ziegler@cs.tu-darmstadt.de

Yannick Loeck
Technische Universität Hamburg
yannick.loeck@tuhh.de

Christian Dietrich
Technische Universität Hamburg
christian.dietrich@tuhh.de

Notes on Data Base Operating

Author: Jim Gray A

Operating Systems, An Ac

Published: 01 January 19

Are Your Database Management System?

Andrew Crotty
Carnegie Mellon University

Viktor Leis
Friedrich-Alexander-Universität

Andy Pavlo
Carnegie Mellon University

Async-fork: Mitigati… Fork-based Snap…

Pu Pang
Shanghai Jiao Tong University
Alibaba Group
avengerispp@sjtu.edu.cn

Operating system support…

Author: Michael Stonebraker Authors Info & Clai…

Communications of the ACM, Volume 24, Issue 7 • pp …

Notes on Data Base Opera…

Author: Jim Gray A…

Operating Systems, An Ac…

Published: 01 January 19…

579 0

Andrew C…
Carnegie Mell…

TYPES OF DATABASE PAPER

…y the …l

…ng System Co-Design

…mie, Adrian Schüpbach

… Management

…MOD 2023

Tobias Ziegler
Technische Universität Darmstadt
tobias.ziegler@cs.tu-darmstadt.de

…ian Dietrich
…niversität Hamburg
…ietrich@tuhh.de

…t System?

Andy Pavlo
Carnegie Mellon University

Case Study: Virtual Memory Snapshotting

Case Study: Virtual Memory Snapshotting

- Redis uses fork to save process memory as checkpoints for persistence

Case Study: Virtual Memory Snapshotting

- Redis uses fork to save process memory as checkpoints for persistence
- Hyper[Kemper et al, ICDE'11] uses fork to run a OLAP queries on a OLTP database
  - Hyper gave up on fork because fork is slow and hard to control

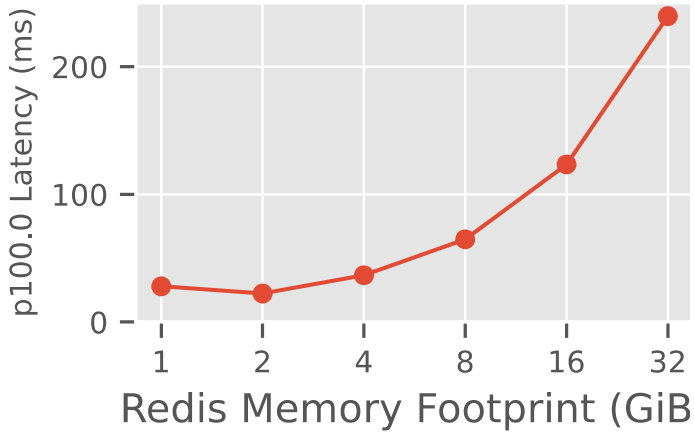
Case Study: Virtual Memory Snapshotting

- Redis uses fork to save process memory as checkpoints for persistence
- Hyper[Kemper et al, ICDE'11] uses fork to run a OLAP queries on a OLTP database
  - Hyper gave up on fork because fork is slow and hard to control
- fork is **blocking** and requires threads to be paused to get a consistent snapshot

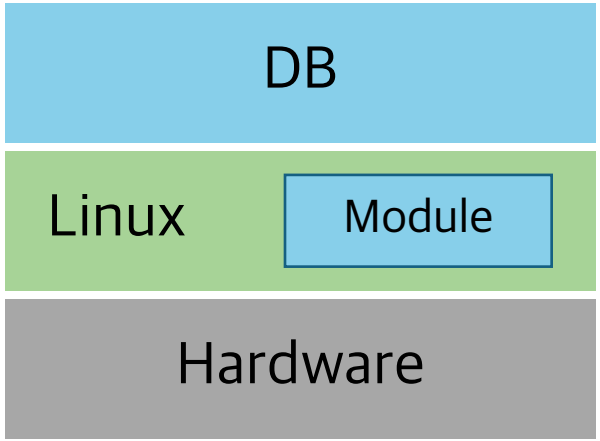
Case Study: Virtual Memory Snapshotting

- Redis uses fork to save process memory as checkpoints for persistence
- Hyper[Kemper et al, ICDE'11] uses fork to run a OLAP queries on a OLTP database
  - Hyper gave up on fork because fork is slow and hard to control
- fork is **blocking** and requires threads to be paused to get a consistent snapshot

Redis p100 Query Latency during Checkpointing

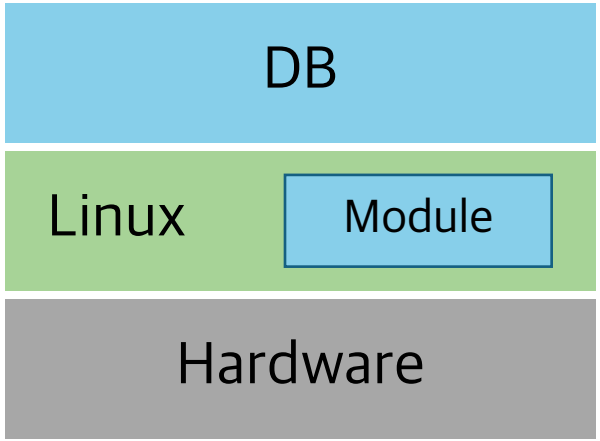


Co-Design Paradigms for this Problem

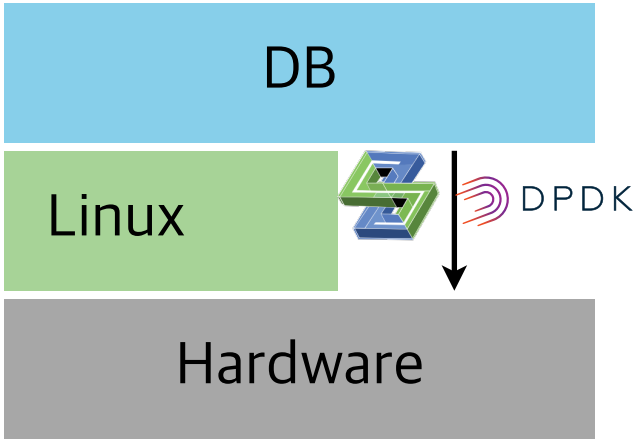


Customized Linux Kernel
- **Security**
- **Maintainability**

Co-Design Paradigms for this Problem

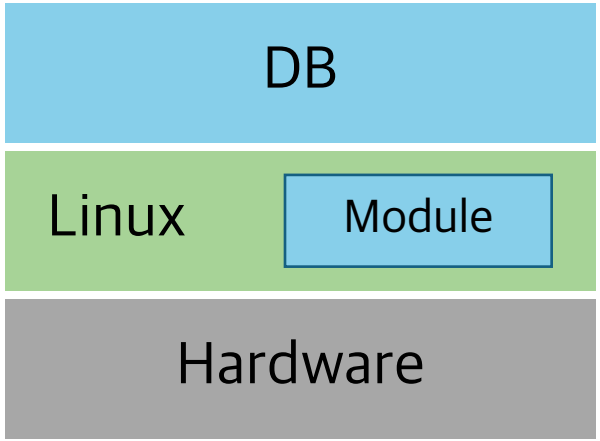


Customized Linux Kernel
- **Security**
- **Maintainability**

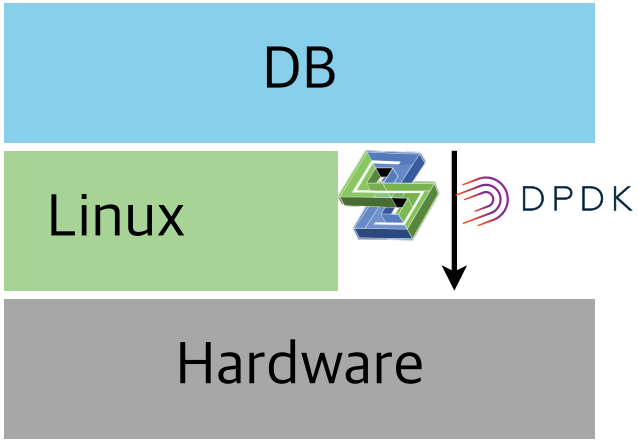


Kernel Bypass
- **No direct control on MMU & Page Table**
- **Limited Design Space**

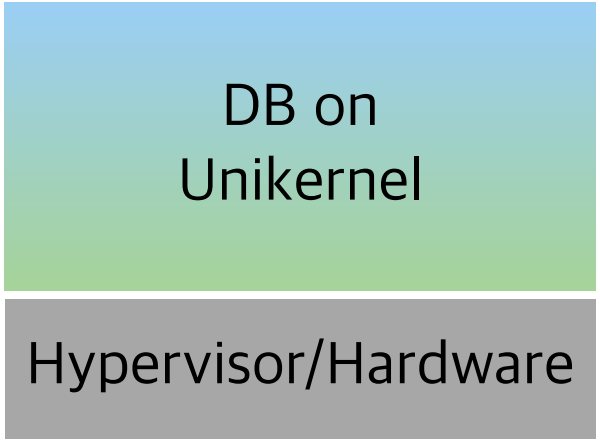
Co-Design Paradigms for this Problem



Customized Linux Kernel
- **Security**
- **Maintainability**



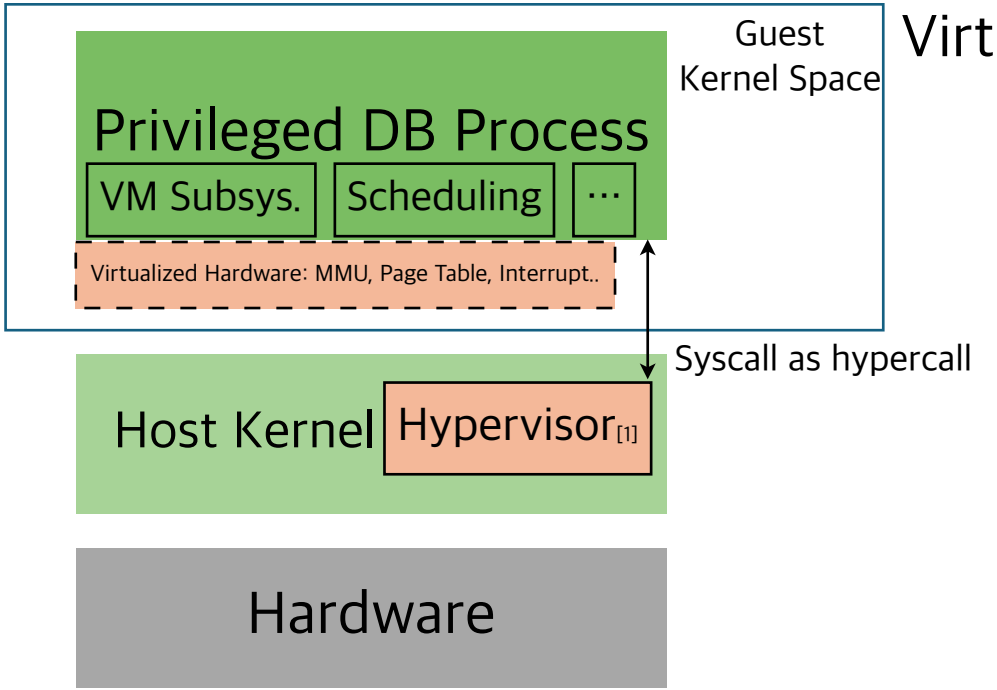
Kernel Bypass
- **No direct control on MMU & Page Table**
- **Limited Design Space**



DB-Unikernel
- **All-or-nothing**
- **Throwing baby(ecosystem) out with bathwater(POSIX)**

How to allow DBMS complete freedom to specialize subsystems while minimizing impact on security, eco-system, and compatibility ?

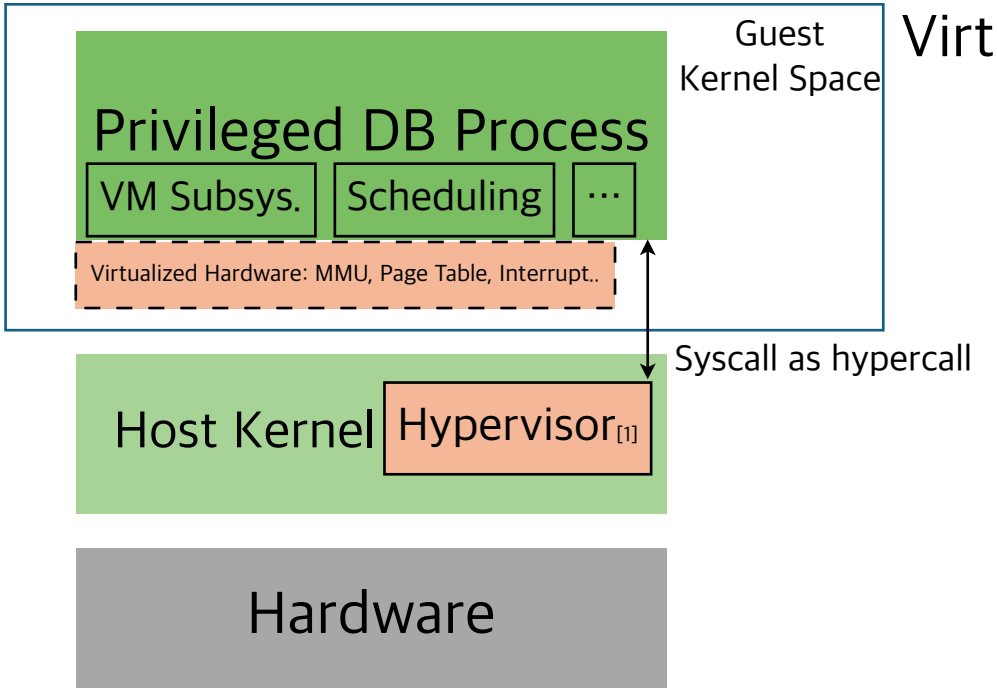
Making DB Process Privileged with Virtualization



Privileged Kernel Bypass

[1] Belay, Adam, et al. "Dune: Safe user-level access to privileged {CPU} features." OSDI 12. 2012.

Making DB Process Privileged with Virtualization



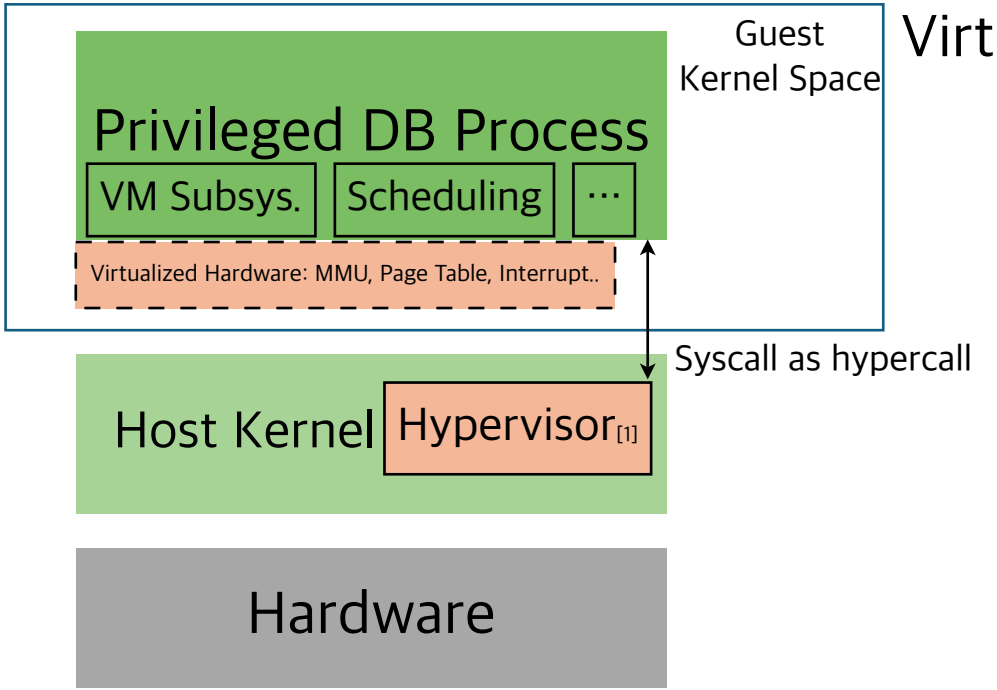
Privileged Kernel Bypass

Virtualized Process

- Selectively specialize security-sensitive subsystems for DBMS to avoid POSIX

[1] Belay, Adam, et al. "Dune: Safe user-level access to privileged {CPU} features." OSDI 12. 2012.

Making DB Process Privileged with Virtualization



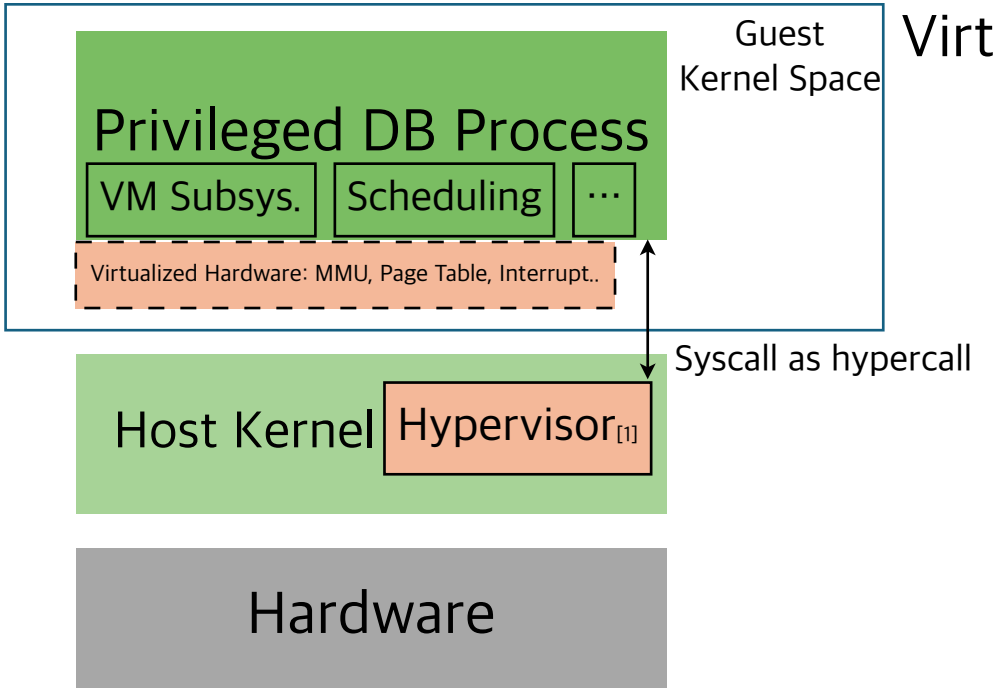
Privileged Kernel Bypass

Virtualized Process

- Selectively specialize security-sensitive subsystems for DBMS to avoid POSIX
- Hypervisor-based isolation for security

[1] Belay, Adam, et al. "Dune: Safe user-level access to privileged {CPU} features." OSDI 12. 2012.

Making DB Process Privileged with Virtualization



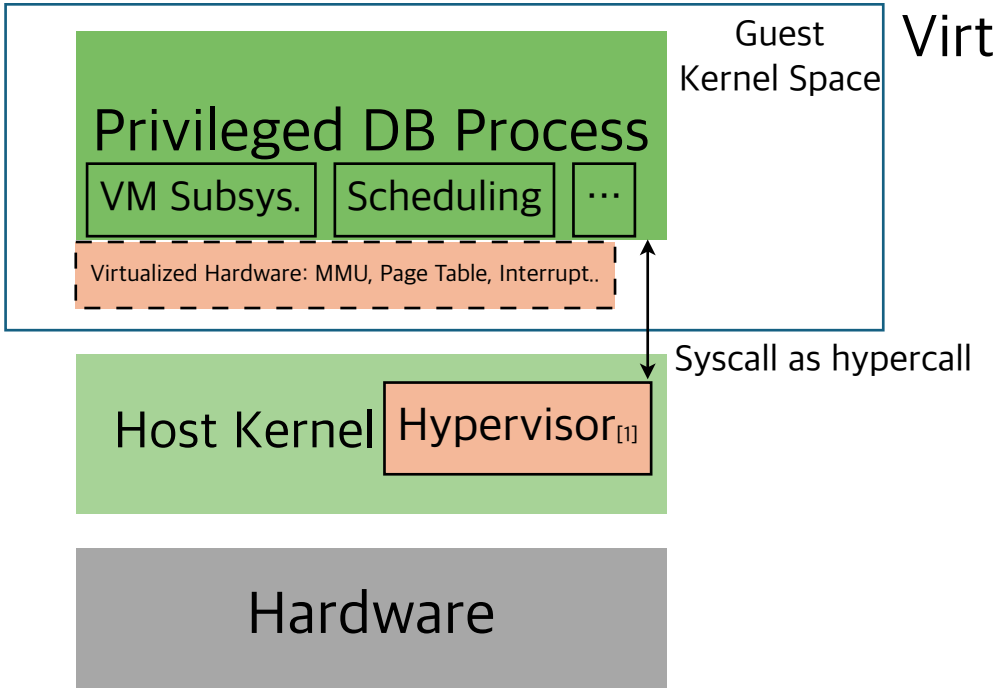
Privileged Kernel Bypass

Virtualized Process

- Selectively specialize security-sensitive subsystems for DBMS to avoid POSIX
- Hypervisor-based isolation for security
- Reuse host kernel functionality and its ecosystem

[1] Belay, Adam, et al. "Dune: Safe user-level access to privileged {CPU} features." OSDI 12. 2012.

Making DB Process Privileged with Virtualization



Privileged Kernel Bypass

Virtualized Process

- Selectively specialize security-sensitive subsystems for DBMS to avoid POSIX
- Hypervisor-based isolation for security
- Reuse host kernel functionality and its ecosystem
- Not throwing the baby out with bathwater

[1] Belay, Adam, et al. "Dune: Safe user-level access to privileged {CPU} features." OSDI 12. 2012.

Privileged Kernel-Bypass vs. Kernel-Bypass for DBMS

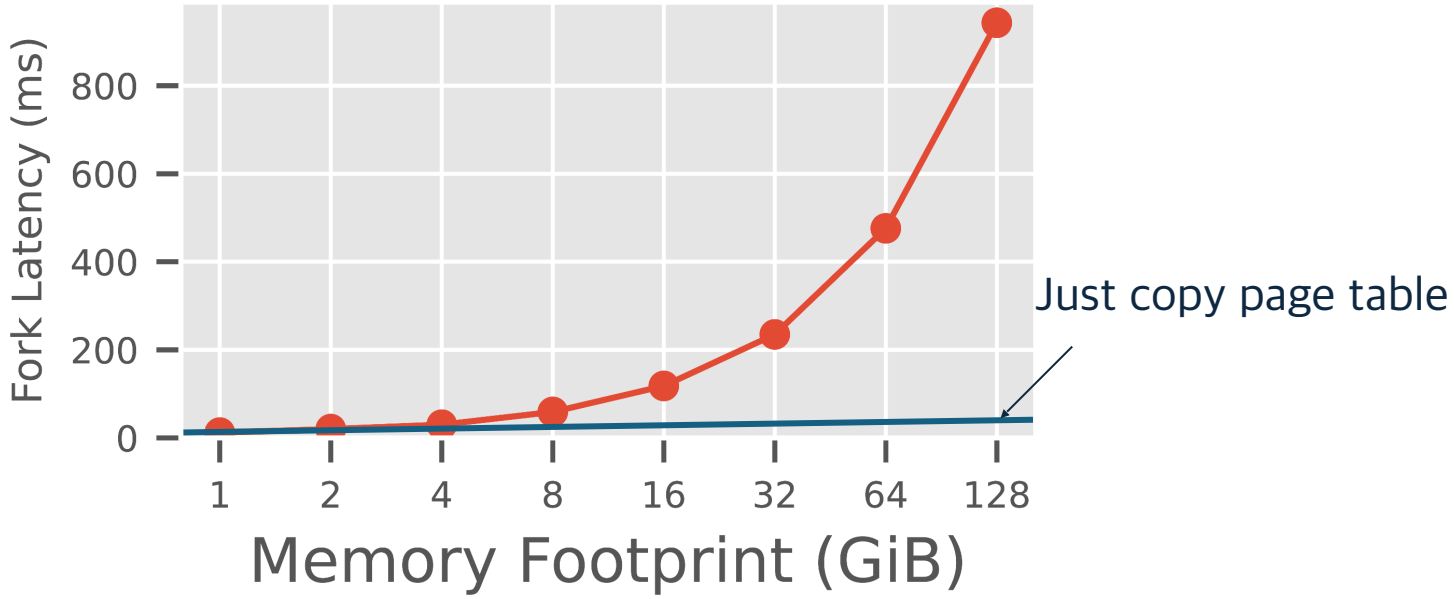
	Kernel Bypass	**Privileged Kernel Bypass**
DBMS runs in	User Space	Guest Kernel Space
Specializes	Network/Storage	Virtual Memory/Scheduler/Interrupt /Network/Storage

This paper

- **Instantaneous snapshotting. ←**
- “Perfect” virtual-memory-assisted buffer manager **[see paper]**

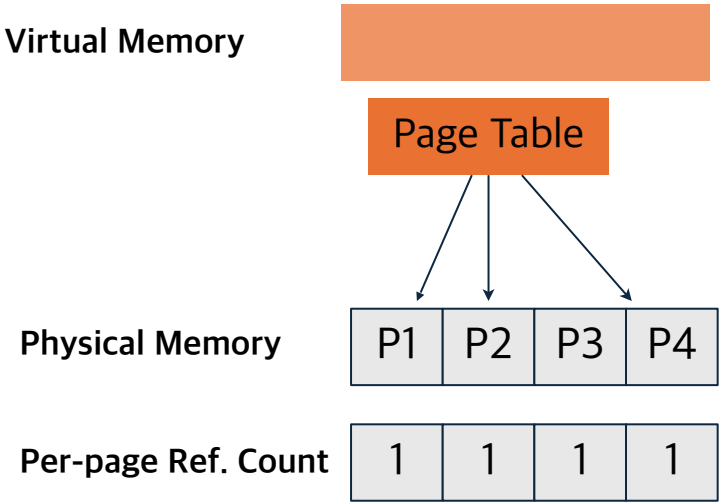
Linux fork Bottleneck Analysis

Linux fork Bottleneck Analysis



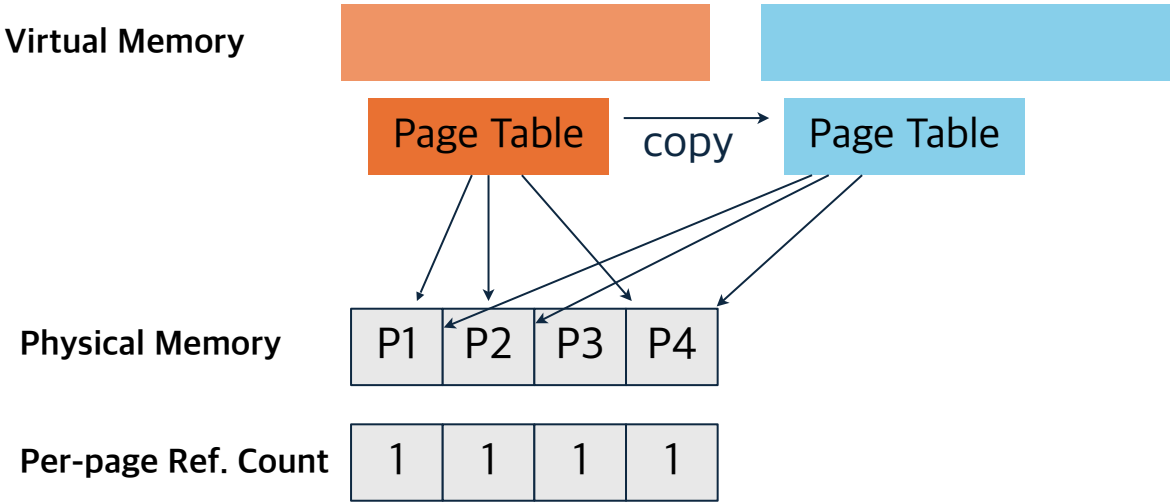
Linux fork Bottleneck Analysis

- Linux kernel maintains a per-page reference count for safe page reclamation - a fundamental design decision to support shared-memory, page cache, memory-mapped files …



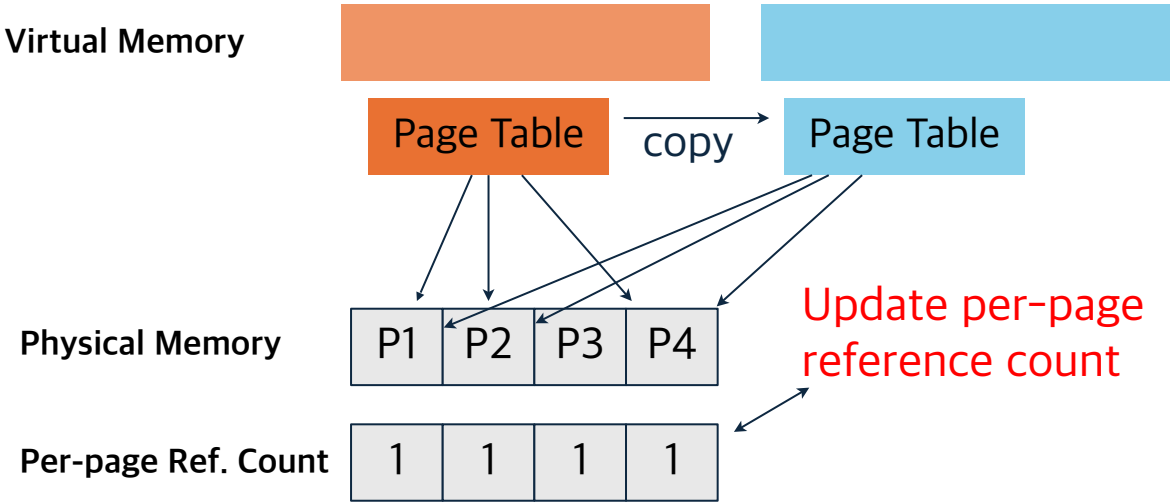
Linux fork Bottleneck Analysis

- Linux kernel maintains a per-page reference count for safe page reclamation - a fundamental design decision to support shared-memory, page cache, memory-mapped files …



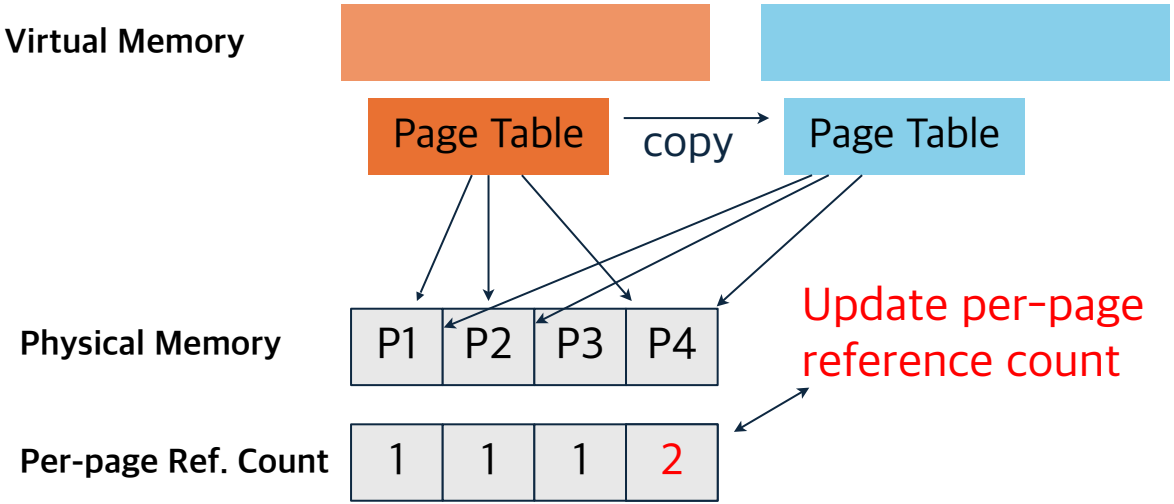
Linux fork Bottleneck Analysis

- Linux kernel maintains a per-page reference count for safe page reclamation - a fundamental design decision to support shared-memory, page cache, memory-mapped files …



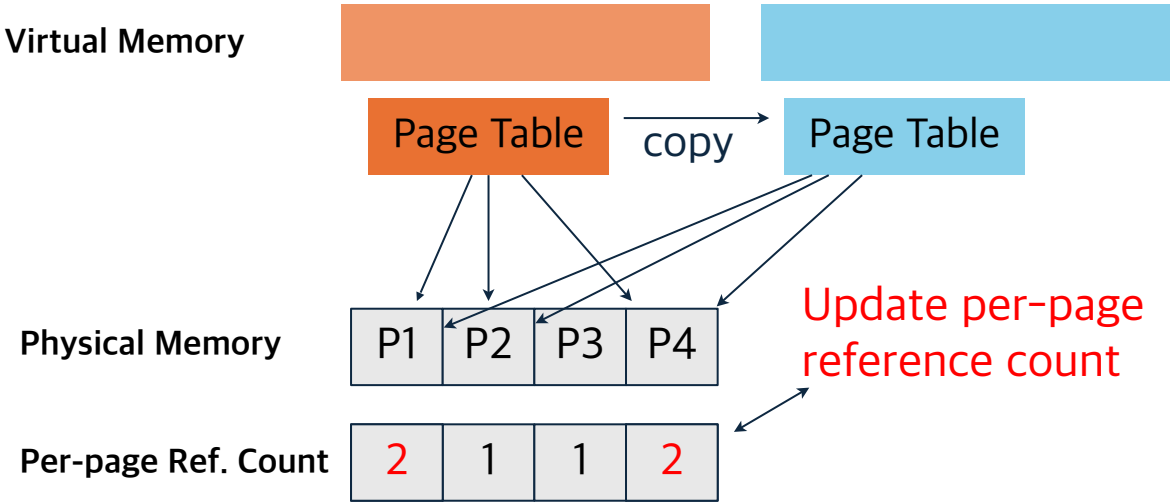
Linux fork Bottleneck Analysis

- Linux kernel maintains a per-page reference count for safe page reclamation - a fundamental design decision to support shared-memory, page cache, memory-mapped files …



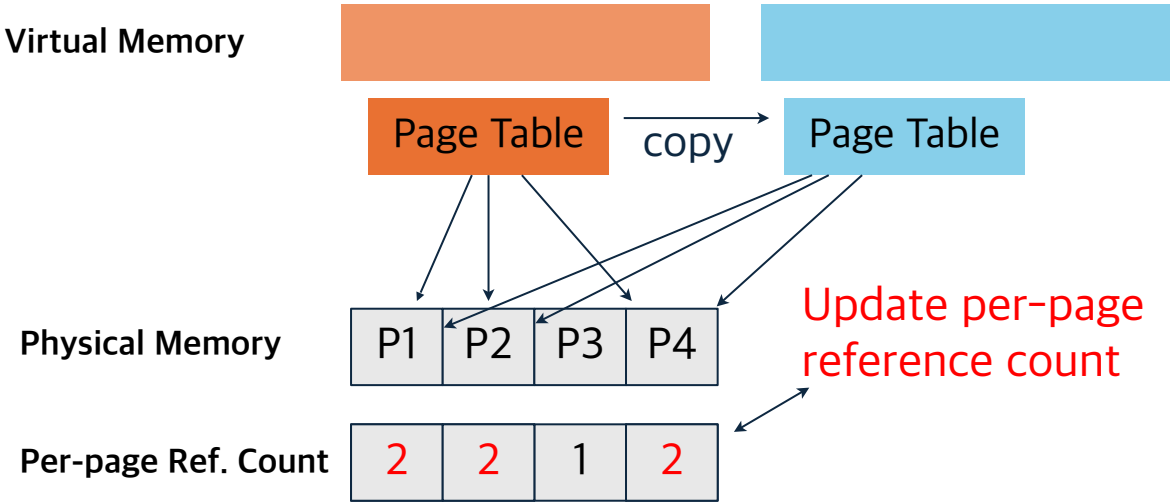
Linux fork Bottleneck Analysis

- Linux kernel maintains a per-page reference count for safe page reclamation - a fundamental design decision to support shared-memory, page cache, memory-mapped files …



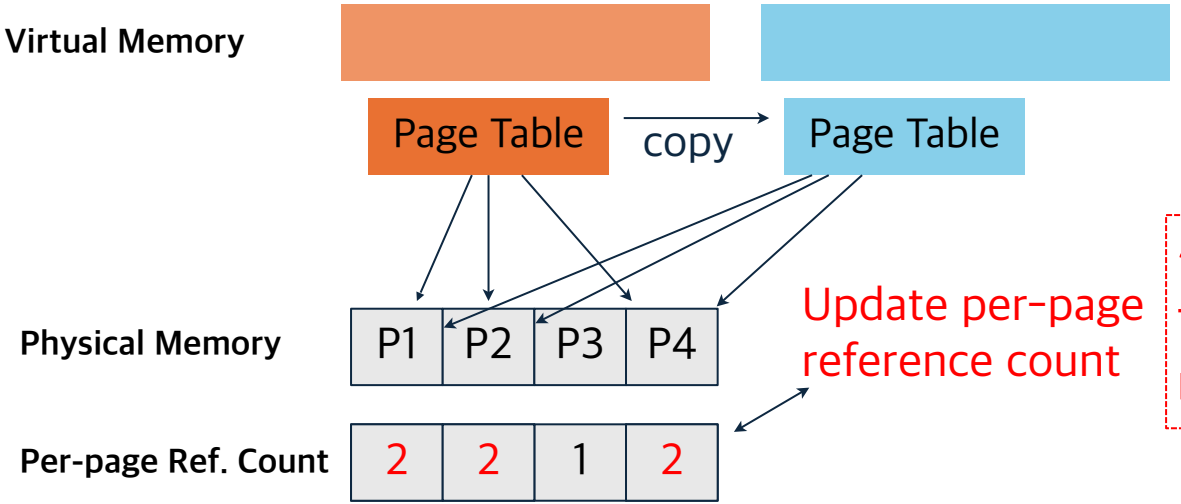
Linux fork Bottleneck Analysis

- Linux kernel maintains a per-page reference count for safe page reclamation - a fundamental design decision to support shared-memory, page cache, memory-mapped files …



Linux fork Bottleneck Analysis

- Linux kernel maintains a per-page reference count for safe page reclamation - a fundamental design decision to support shared-memory, page cache, memory-mapped files …



~90% CPU cycles due to cache misses and random accesses

Fast Snapshotting with Privileged Kernel Bypass

- We can make many simplifying assumptions
- Specialize a simple VM/snapshotting system in the privileged DB process
  - No reference counting for physical pages – DB is the only user
  - No support for shared-memory, page cache, memory-mapped files…
  - No nested snapshot – Redis/KeyDB/Hyper use cases
- Challenge: how to safely reclaim physical pages without reference count?

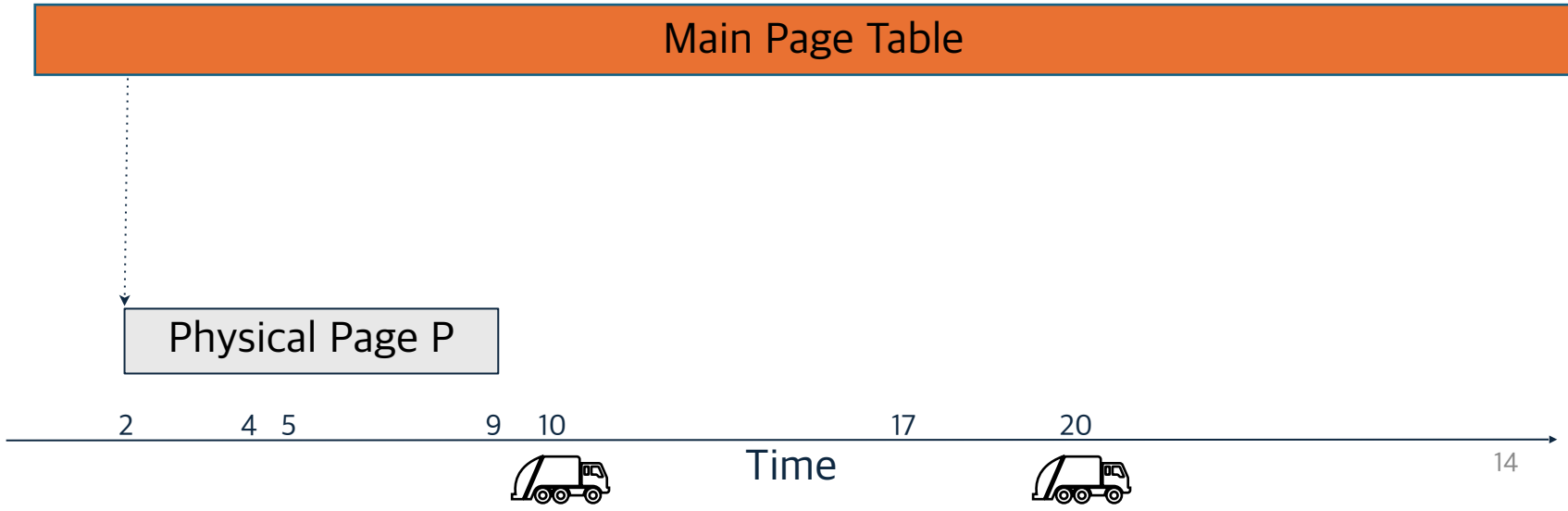
Safely Reclaiming Physical Pages with Timestamp

- Lifecycle of page/snapshot tracked with timestamps – akin to epoch-based reclamation
- A page is reclaimed when there are no references from any page tables.
  - No overlap between the lifetime of active page tables and physical page
- Pages are periodically examined for garbage collection in batches

Main Page Table

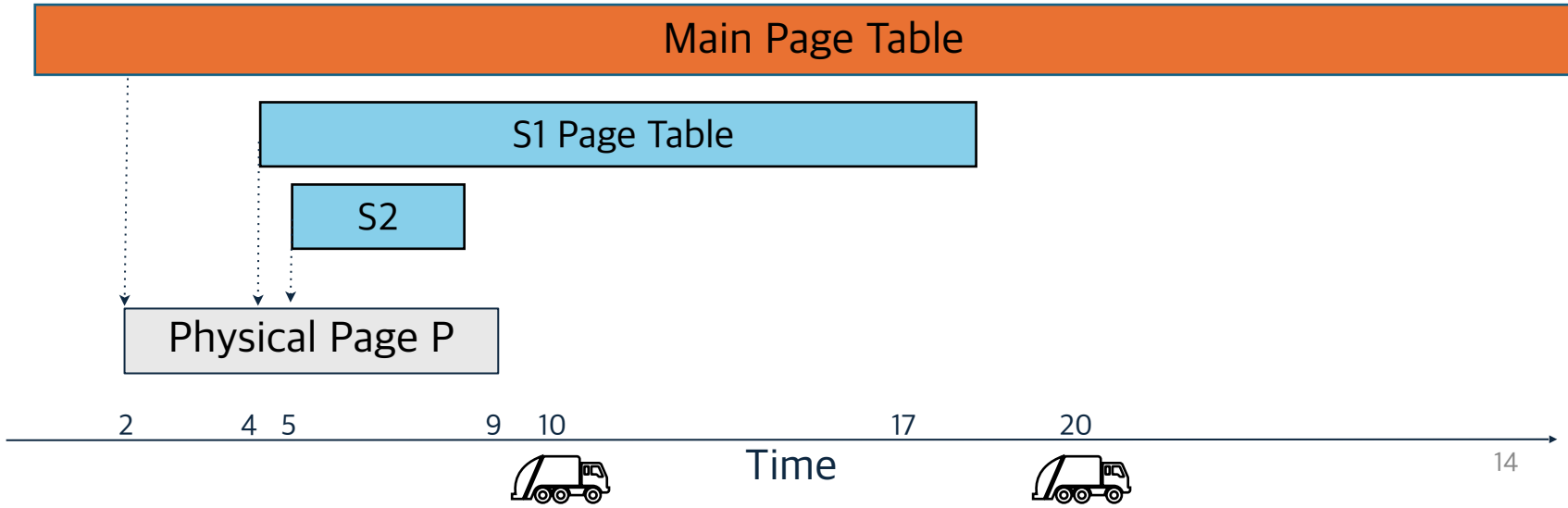
Safely Reclaiming Physical Pages with Timestamp

- Lifecycle of page/snapshot tracked with timestamps – akin to epoch-based reclamation
- A page is reclaimed when there are no references from any page tables.
  - No overlap between the lifetime of active page tables and physical page
- Pages are periodically examined for garbage collection in batches



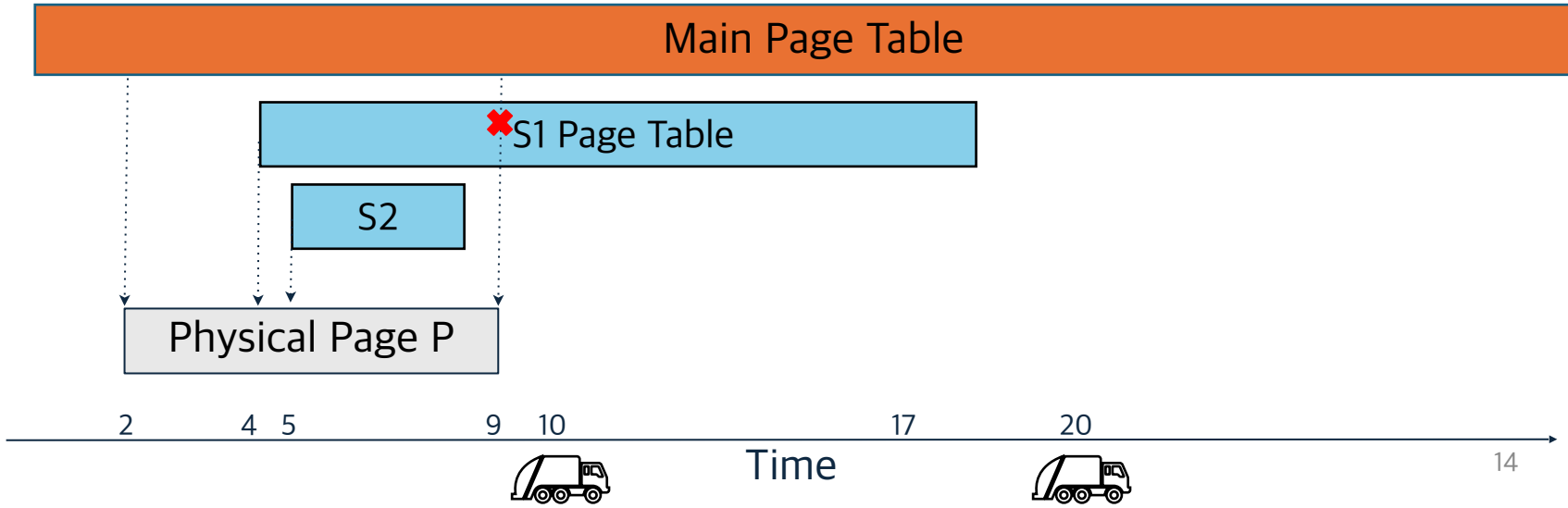
Safely Reclaiming Physical Pages with Timestamp

- Lifecycle of page/snapshot tracked with timestamps – akin to epoch-based reclamation
- A page is reclaimed when there are no references from any page tables.
  - No overlap between the lifetime of active page tables and physical page
- Pages are periodically examined for garbage collection in batches



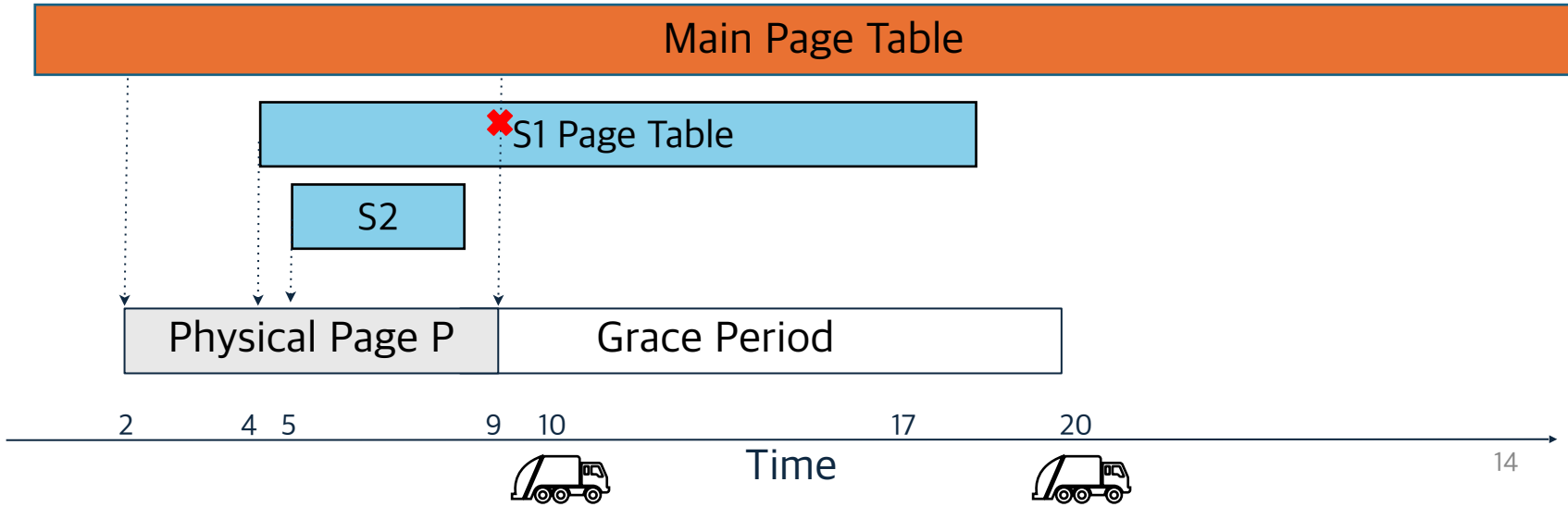
Safely Reclaiming Physical Pages with Timestamp

- Lifecycle of page/snapshot tracked with timestamps – akin to epoch-based reclamation
- A page is reclaimed when there are no references from any page tables.
  - No overlap between the lifetime of active page tables and physical page
- Pages are periodically examined for garbage collection in batches



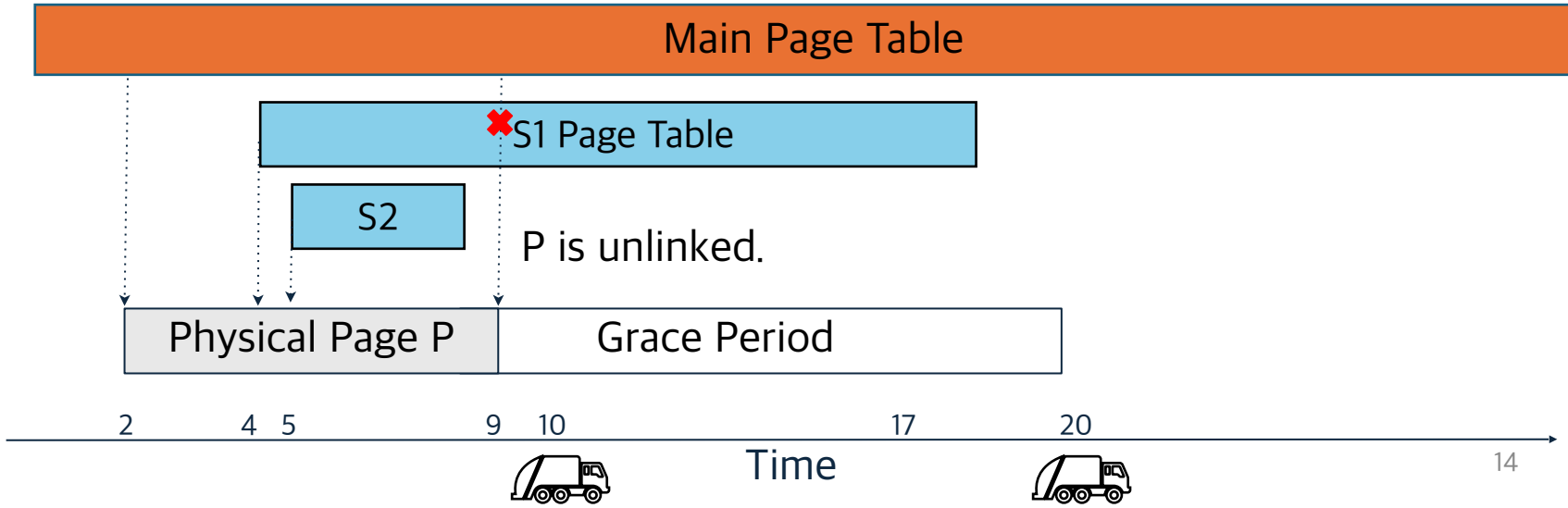
Safely Reclaiming Physical Pages with Timestamp

- Lifecycle of page/snapshot tracked with timestamps – akin to epoch-based reclamation
- A page is reclaimed when there are no references from any page tables.
  - No overlap between the lifetime of active page tables and physical page
- Pages are periodically examined for garbage collection in batches



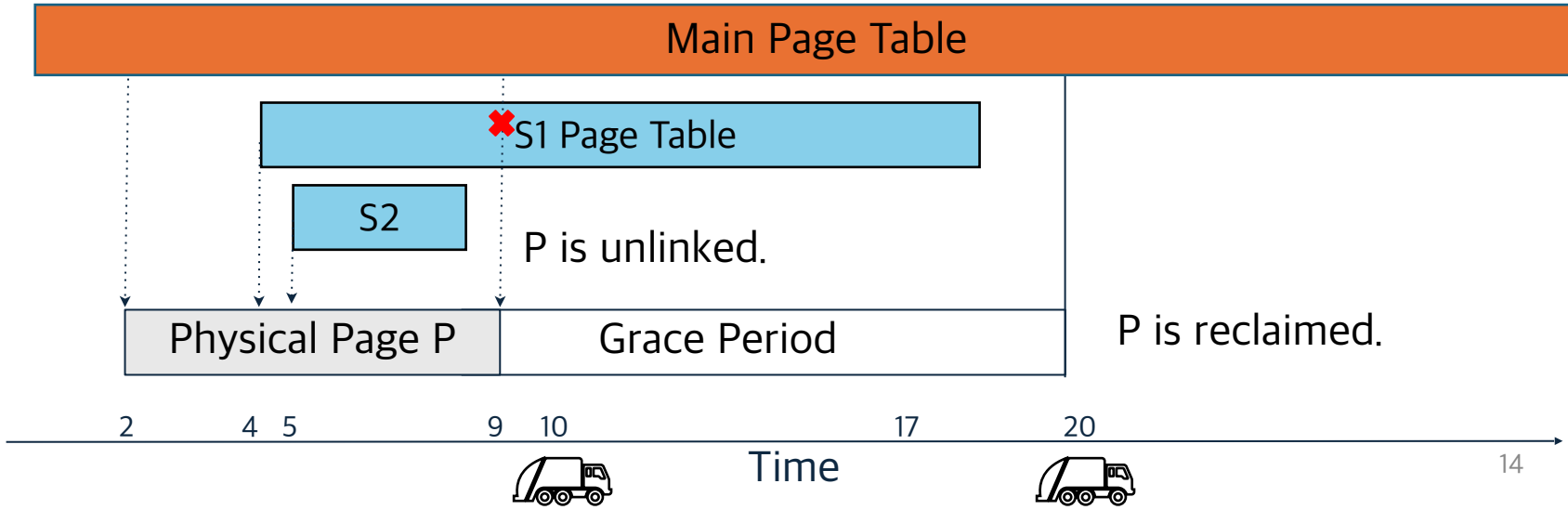
Safely Reclaiming Physical Pages with Timestamp

- Lifecycle of page/snapshot tracked with timestamps – akin to epoch-based reclamation
- A page is reclaimed when there are no references from any page tables.
  - No overlap between the lifetime of active page tables and physical page
- Pages are periodically examined for garbage collection in batches



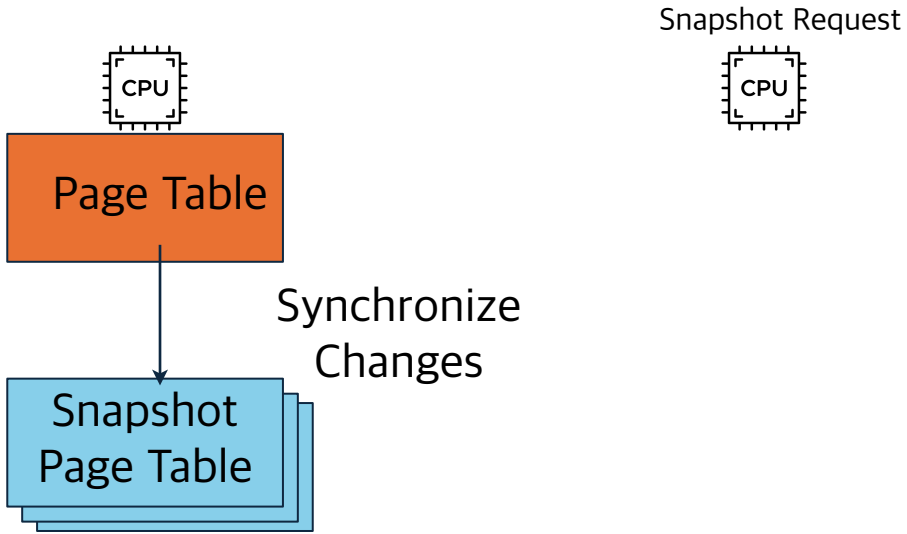
Safely Reclaiming Physical Pages with Timestamp

- Lifecycle of page/snapshot tracked with timestamps – akin to epoch-based reclamation
- A page is reclaimed when there are no references from any page tables.
  - No overlap between the lifetime of active page tables and physical page
- Pages are periodically examined for garbage collection in batches



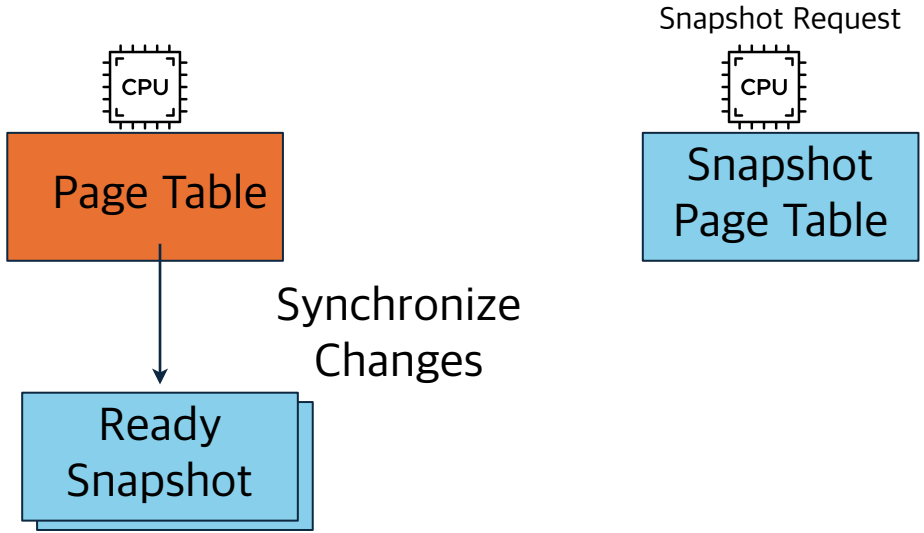
Instant Snapshotting via Pre-Creation

- Asynchronously maintain a set of ready-to-go snapshot page tables
- Completely hide the copy latency, making the snapshot creation appear instant



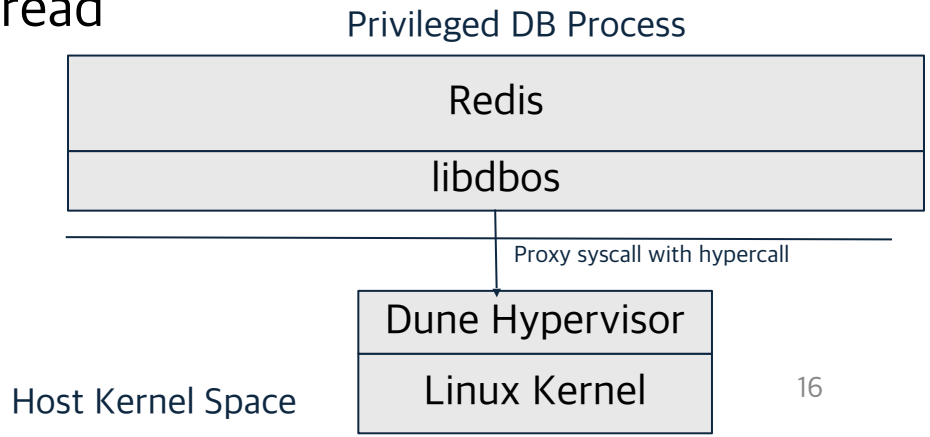
Instant Snapshotting via Pre-Creation

- Asynchronously maintain a set of ready-to-go snapshot page tables
- Completely hide the copy latency, making the snapshot creation appear instant



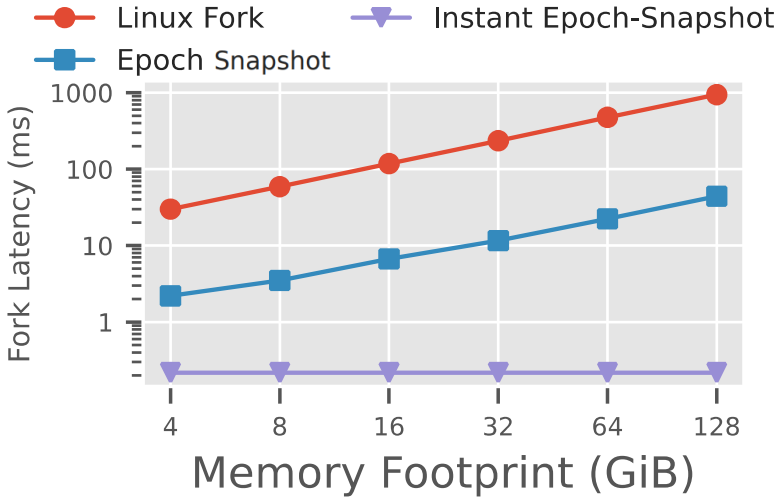
Implementation

- The snapshot mechanism is implemented (~1K LOC) in a guest kernel called **libdbos** on top of Dune hypervisor
  - Linux virtual memory subsystem 110K LOC
- Physical memory backing and system call proxy are done by the hypervisor
- Evaluated on Redis by replacing fork with this snapshot mechanism
  - Checkpoint process runs in a thread

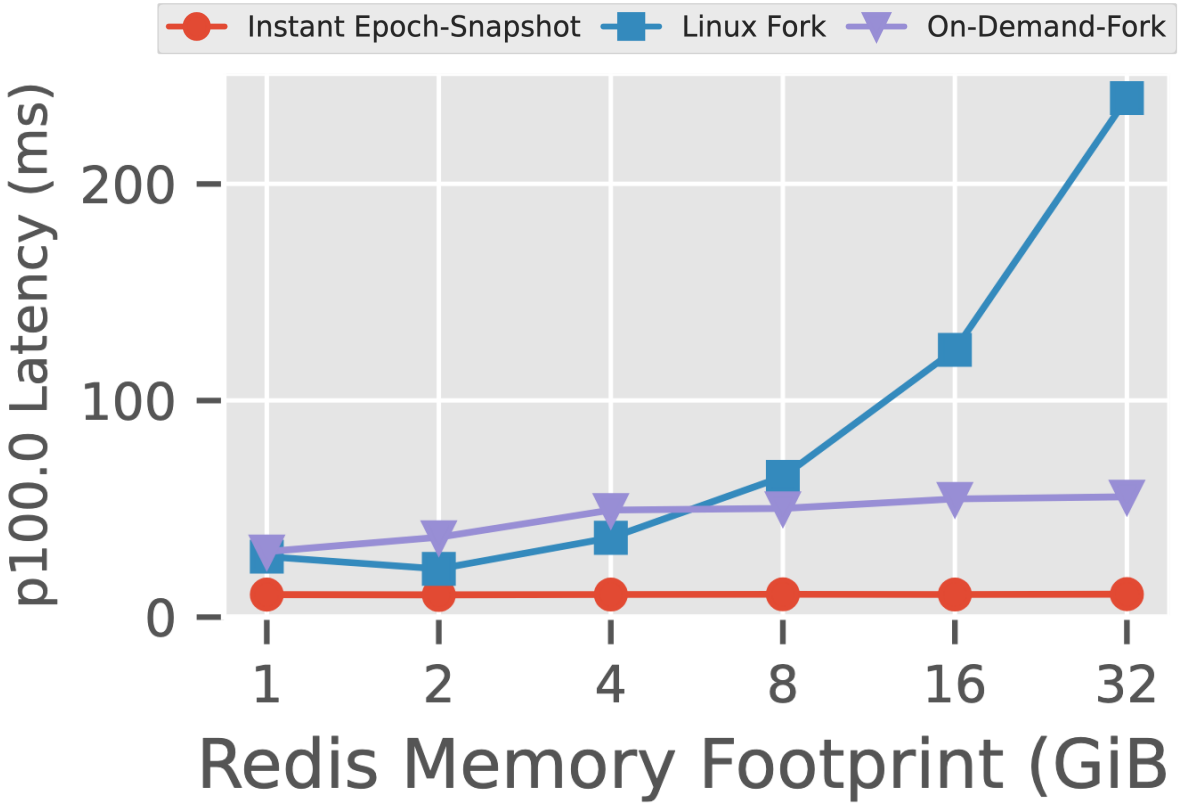


Microbenchmark

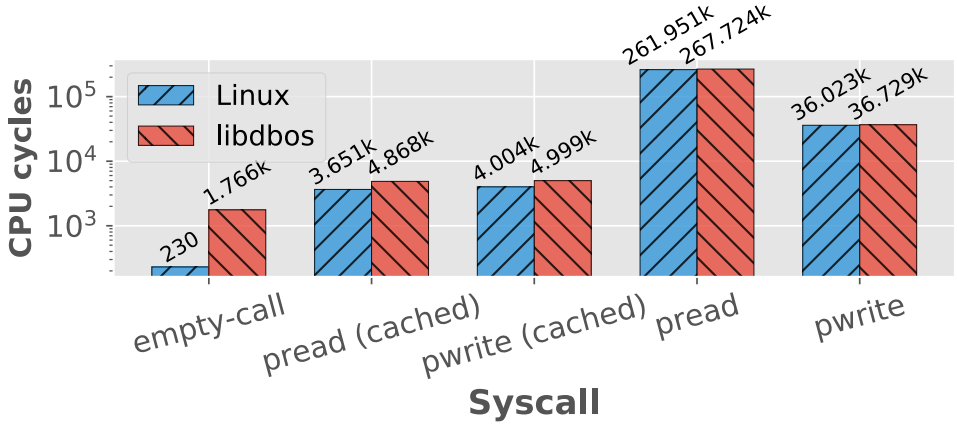
- ~20x reduction in snapshot latency
  - Snapshot 128GB memory in 40ms without parallelization
- Async copy completely hides fork latency if snapshot frequency > page table copy time



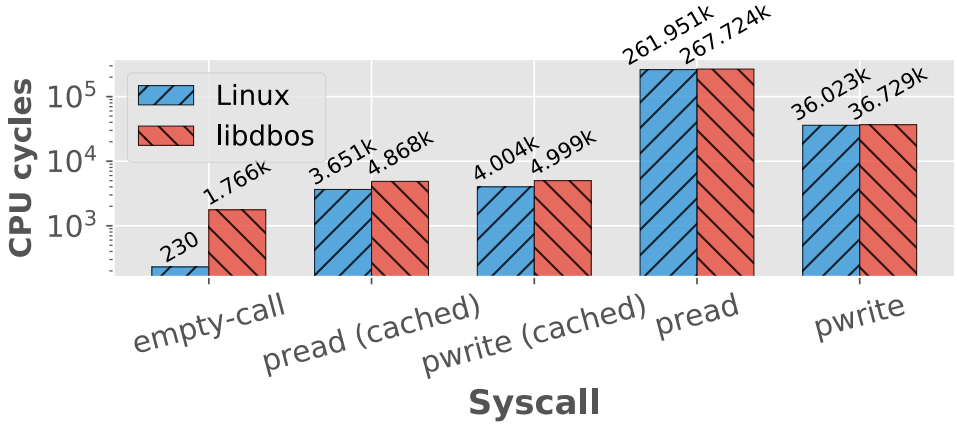
Tail Latency of Redis **SET** Query during Checkpoint

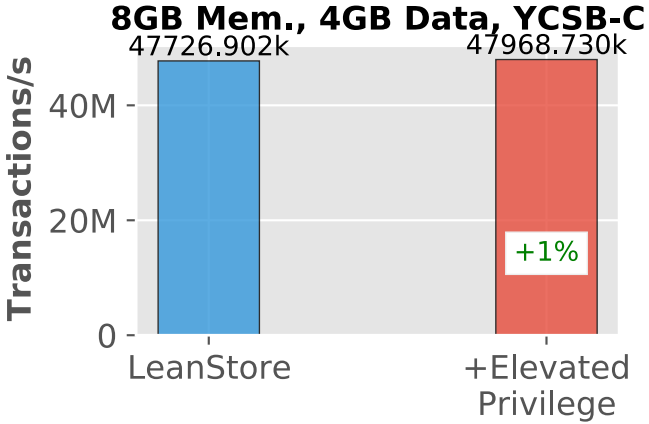


Cost of Virtualization

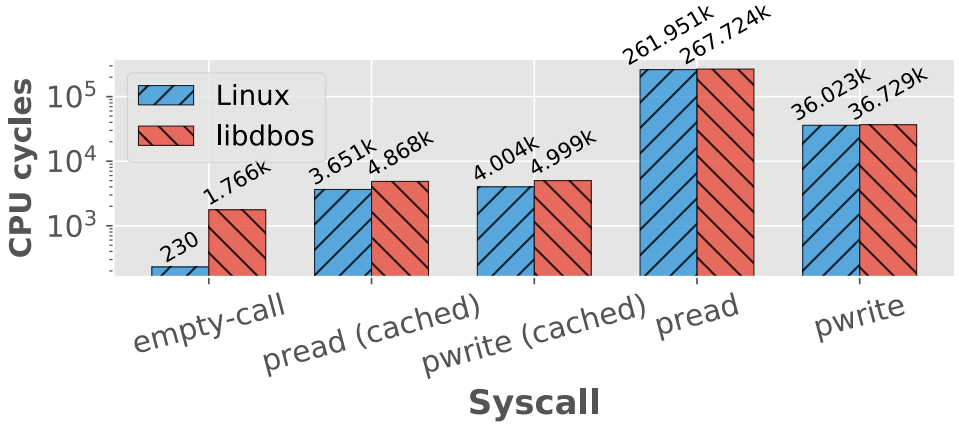


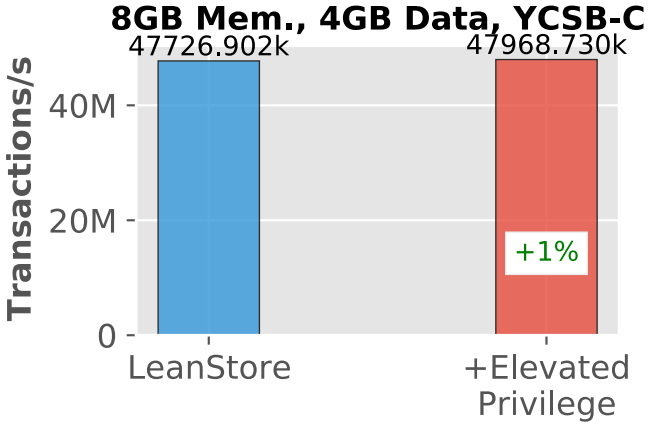
Cost of Virtualization

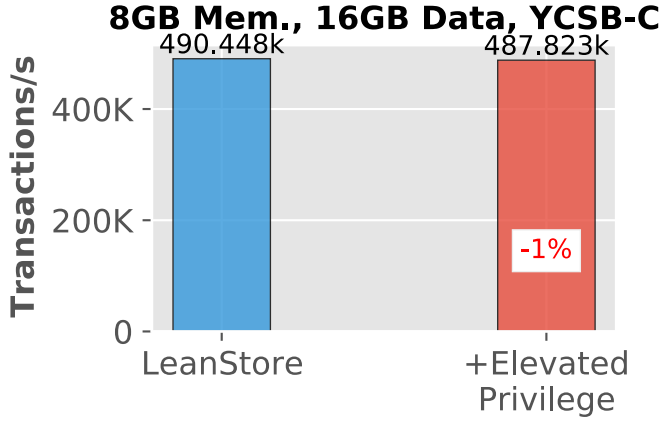




Cost of Virtualization



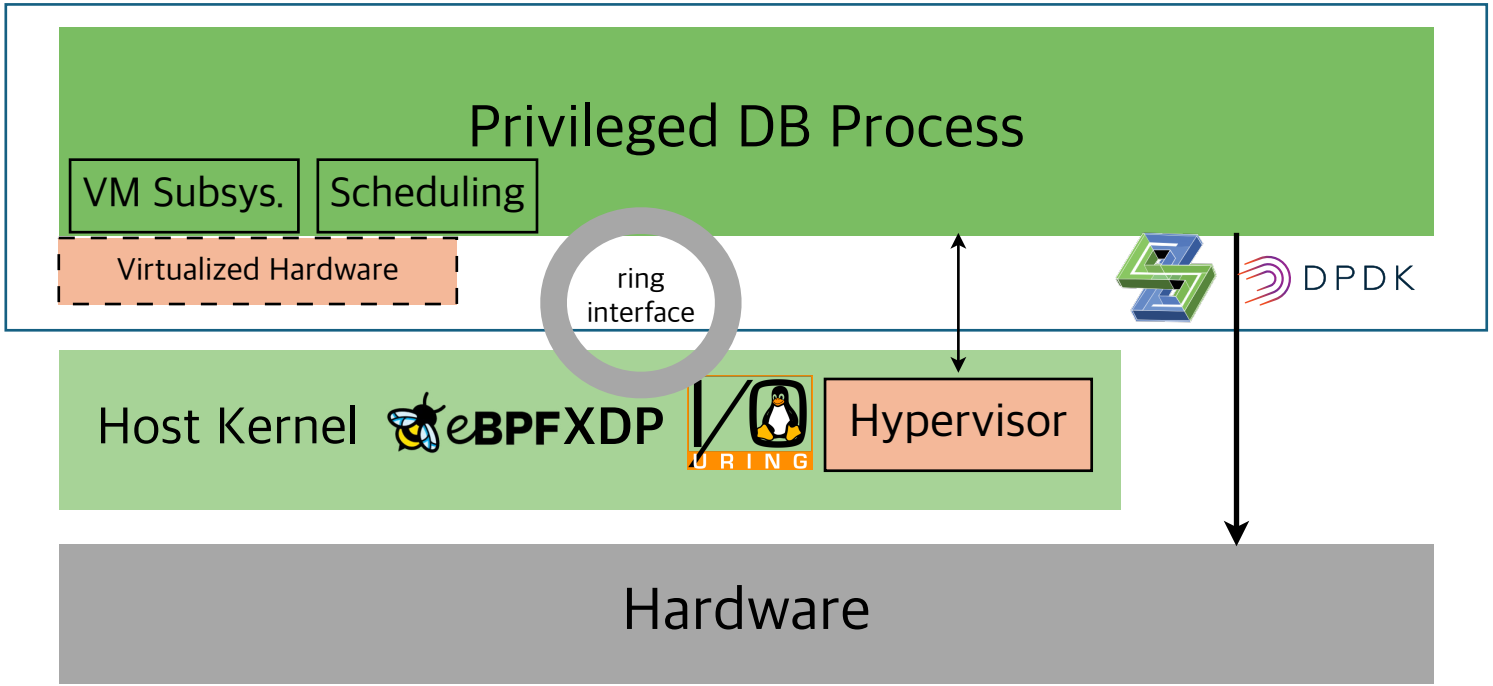




Numerous Possibilities

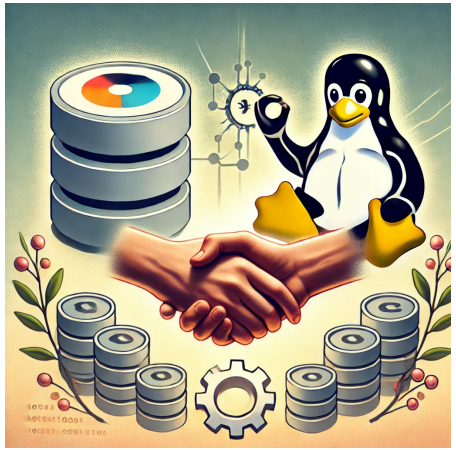
- Virtual Memory
  - “Perfect” virtual-memory-assisted buffer manager
  - Faster memory-rewiring for DBMS query processing and indexing
  - Faster memory allocation
  - …
- Scheduling
  - Robust lightweight task scheduling with preemption
  - Transaction-priority-aware lightweight task scheduling
  - …
- Hypervisor Interface
  - DBMS-assisted memory ballooning

Compatible with Modern Linux Data-Path Interfaces



Conclusions

- With **privileged kernel-bypass**, we can address the mismatch problem while
  - minimizing impact on kernel security and stability
  - providing complete design freedom to DBMS
  - preserving ecosystem
- DBMS deserves to be to **privileged!**



Paper