Mitosis - Transparently Self-Replicating Page-Tables for Large-Memory Machines

Reto Achermann[1]

Ashish Panwar[2]

Abhishek Bhattacharjee[3]

Timothy Roscoe[1]

Jayneel Gandhi[4]

[1]ETH Zurich, [2]IISc Bangalore,
[3]Yale University, [4]VMware Research

ASPLOS '20

vmware

Mitosis: Mitigate NUMA Effects on Page Table Walks

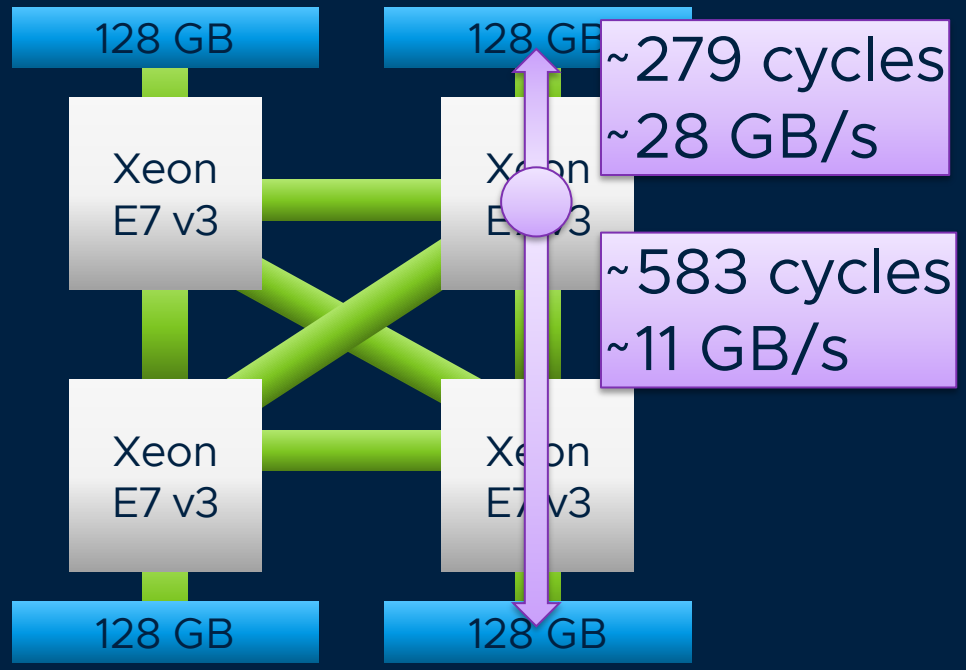
Up to 3.2x speed up for single threaded applications.

Up to 1.3x speed up on multi-threaded applications.

No modifications of the application workloads.

Measured on 4x12 Intel Xeon E7-4850 v3, with 512GB RAM (4x128GB)

NUMA Effects on Large Multi-Socket Machines



Bandwidth & capacity limited per processor socket

More bandwidth & capacity needed
→ Multi-socket machine

Non-uniform memory access (NUMA) machine

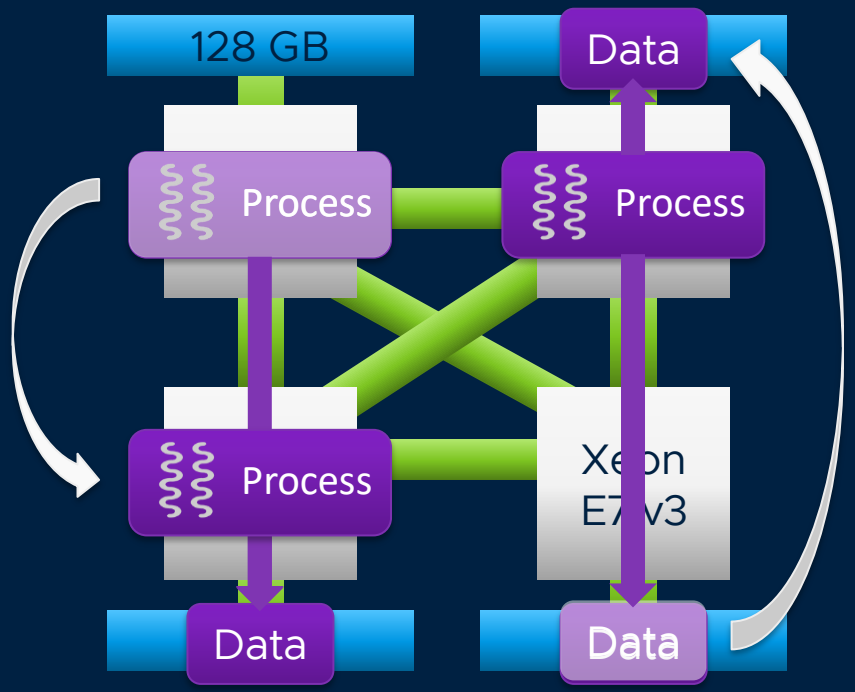
Measured on 4x12 Intel Xeon E7-4850 v3, with 512GB RAM (4x128GB)

vmware®

Handling NUMA Effects on Data Pages

Goal: Keep process threads and data close

NUMA-aware scheduling



NUMA-aware data migration

Decades of work has focused on **data placement**.

What about placement of **page tables?**

Every Memory Access Requires Virtual-to-Physical Translation

Background (x86 specific)

Processes deal with virtual addresses

0x40000 0x930

47 12 11 0

Virtual Page Number | Virtual Page Offset

Translation (MMU)

On every memory access !

45 12 11 0

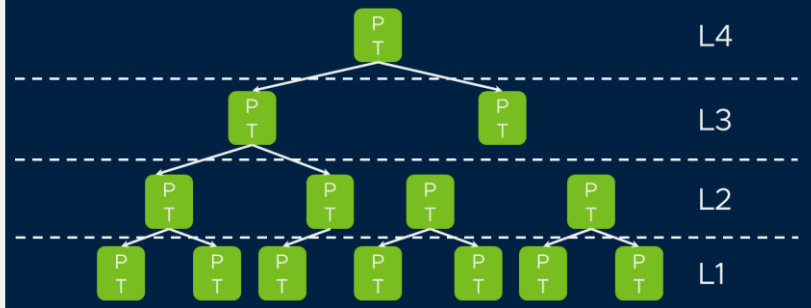
Physical Frame Number | Physical Frame Offset

0x2000 0x930

0x2000930 → RAM

memory knows physical addresses

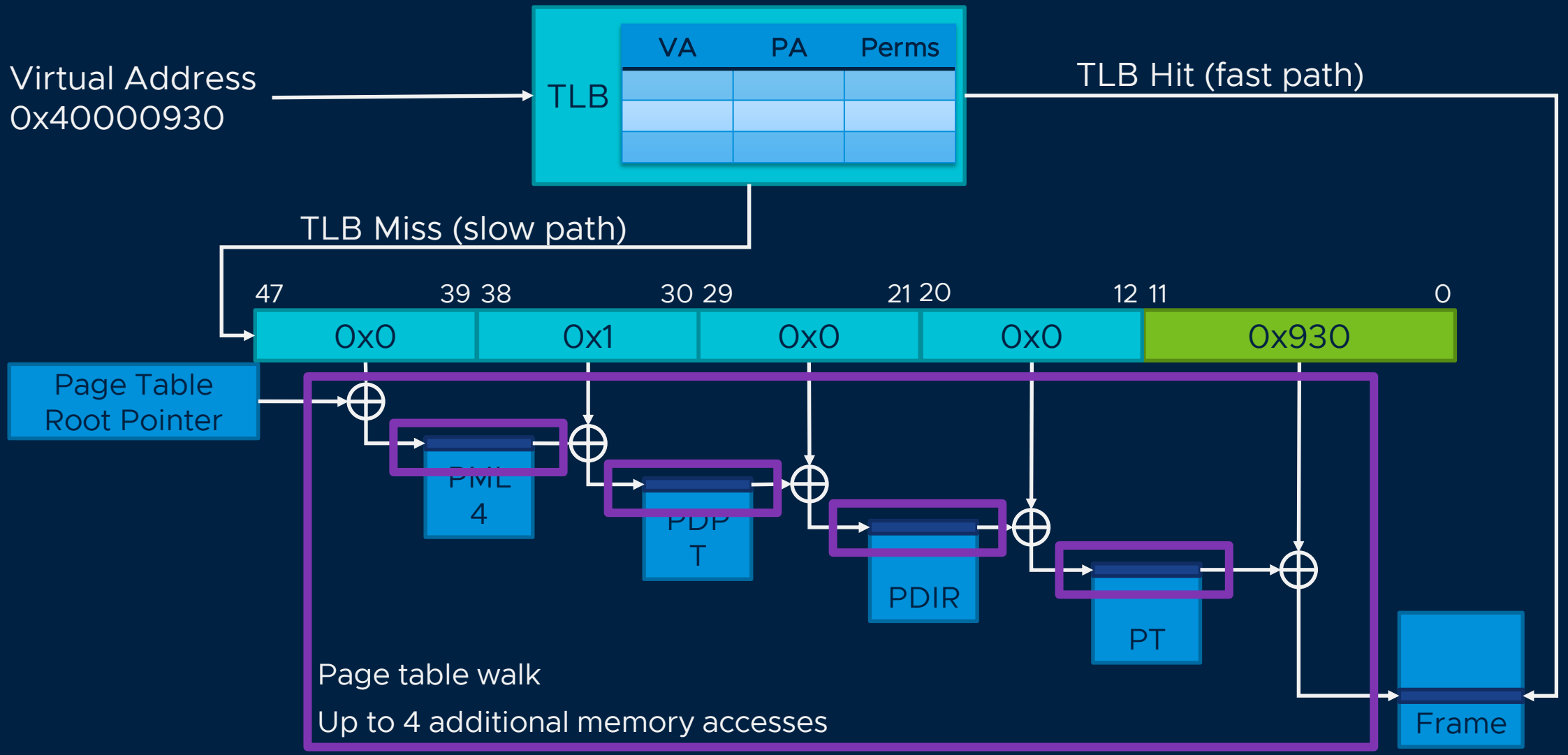
Defined by the page table.



Multi-level radix-tree.

Translation Lookaside Buffer

TLB misses trigger page table walks



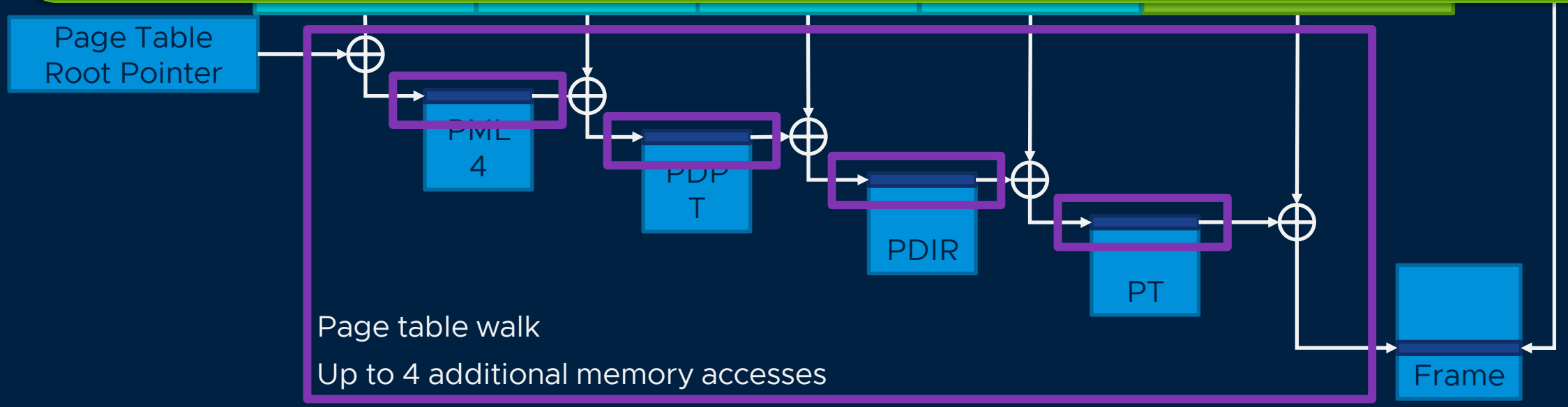
Translation Lookaside Buffer

TLB misses trigger page table walks

Key Problem:

1. TLB coverage << available memory → more frequent TLB misses.
   Test machine with 512GB RAM, TLB coverage < 1% of memory

2. These TLB misses require multiple memory accesses



NUMA effects on page table walks?

Study: Page Table Walks and NUMA

How often is the page table remote and what's the resulting slowdown?

Multi-Threaded Scenario

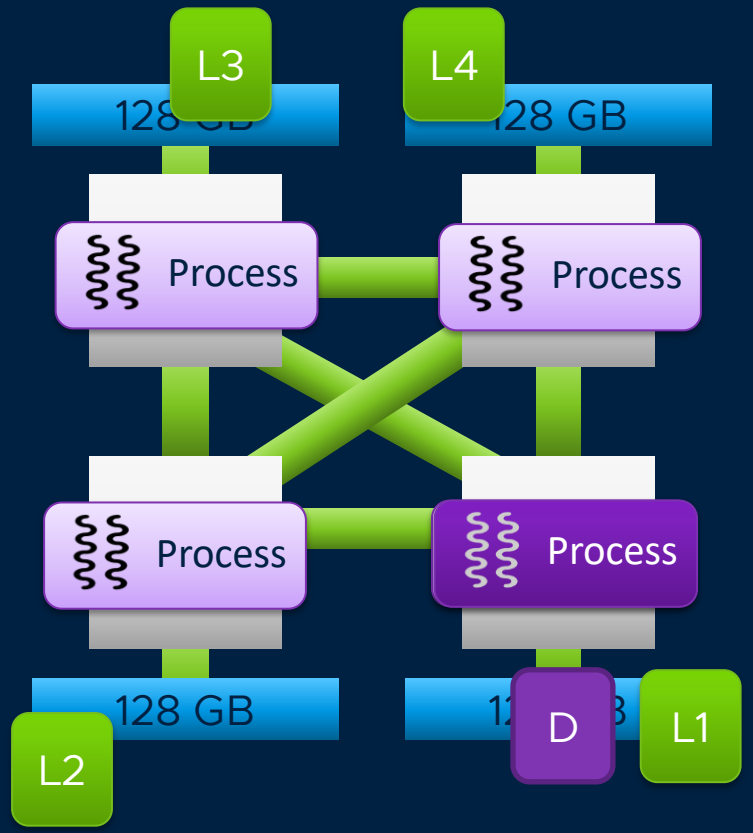
Distribution of page table entries

- Local NUMA Node
- Remote NUMA Node

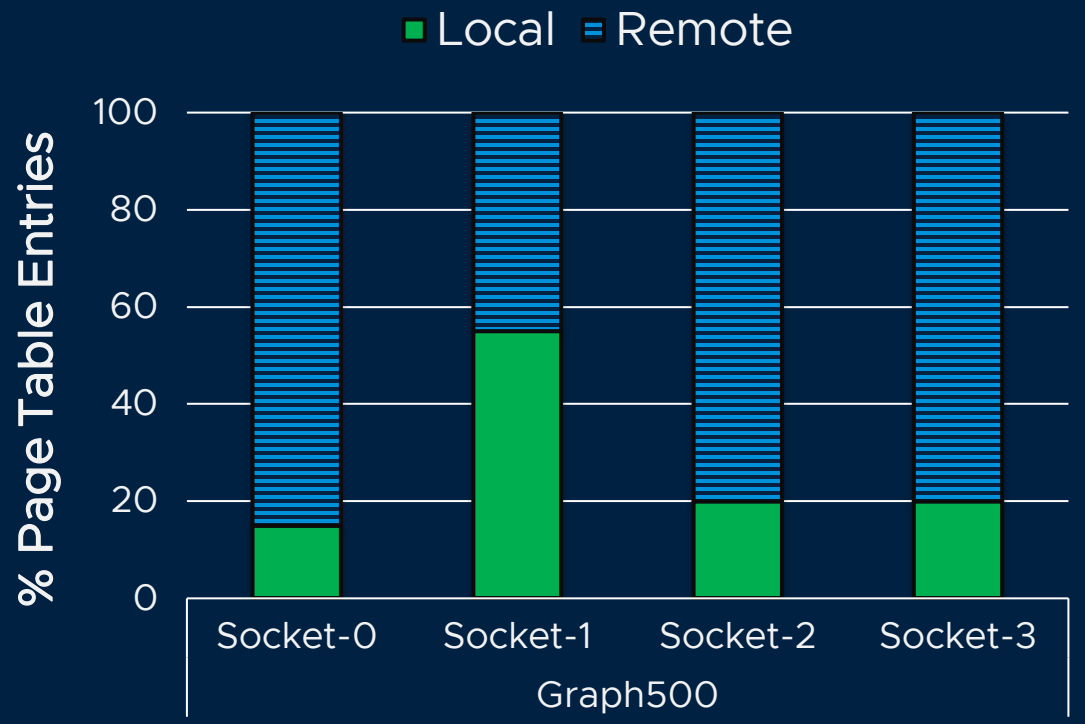
Many Threads – One Page Table

Remote Memory Accesses are Inevitable

More workloads in the paper



Majority of TLB misses require remote memory accesses



Measured on Linux v4.17

vmware

Study: Page Table Walks and NUMA

How often is the page table remote and what's the resulting slowdown?

Multi-Threaded Scenario

Distribution of Page Table Entries

- Local NUMA Node
- Remote NUMA Node

Majority of TLB misses require remote memory accesses

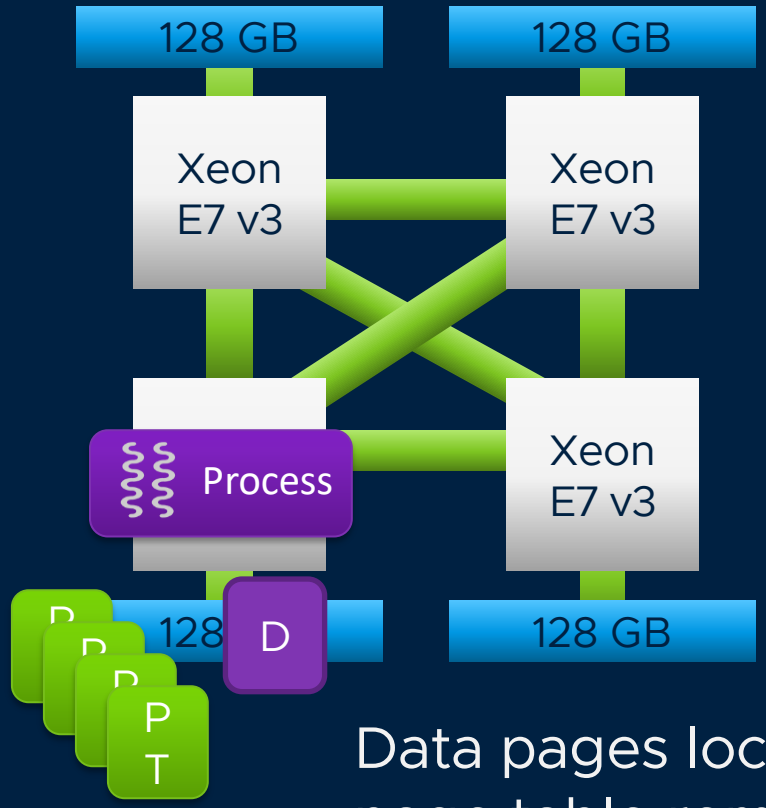
Workload-Migration Scenario

Slow down due to remote memory accesses during page table walks

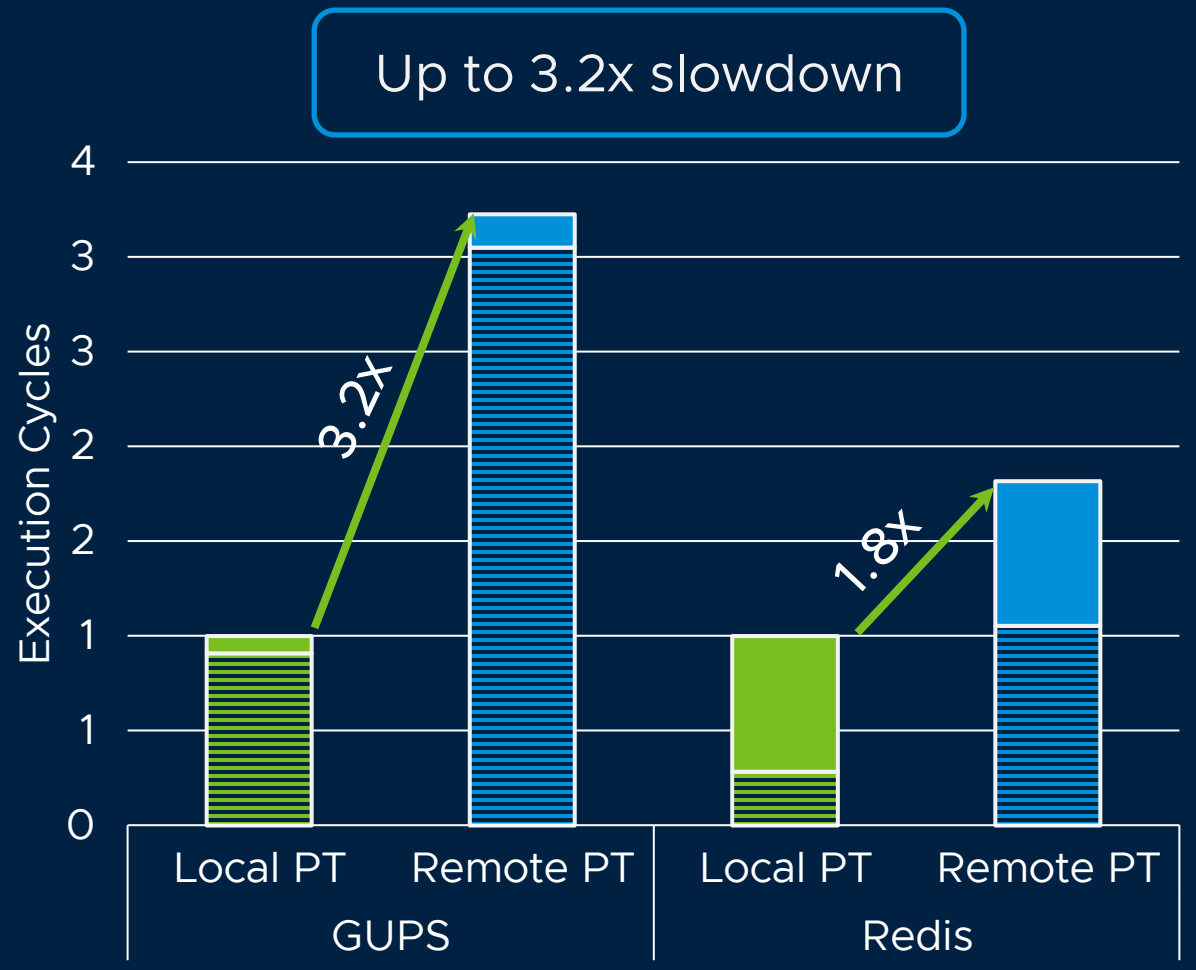
Remote Page Walk Slow Down

Data Migration Policy moves Data Pages, but not Page Tables

More workloads in the paper



Data pages local,
page table remote.



Measured on Linux v4.17

vmware

Study: Page Table Walks and NUMA

How often is the page table remote and what's the resulting slowdown?

Multi-Threaded Scenario

Distribution of Page Table Entries

- Local NUMA Node
- Remote NUMA Node

Majority of TLB misses require remote memory accesses

Workload-Migration Scenario

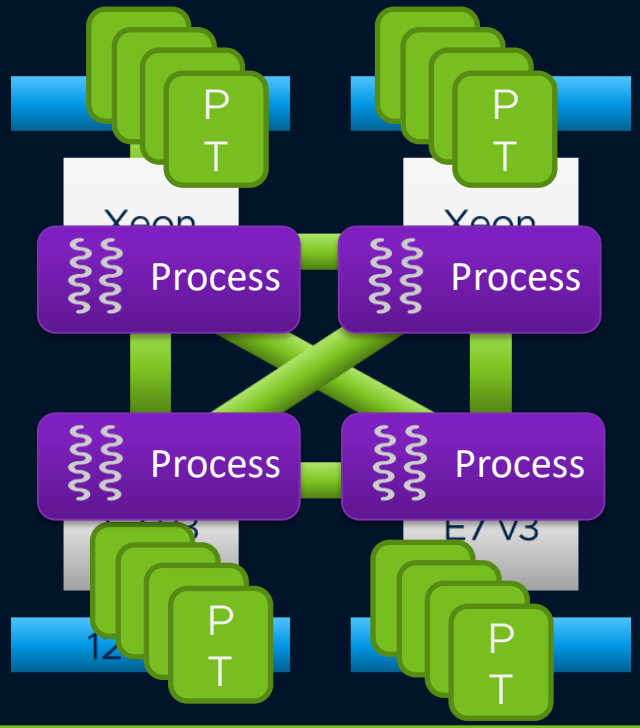
Slow down due to remote memory accesses during page table walks

Up to 3.2x slowdown

Mitosis: Transparent Replication and Migration for Page-Tables

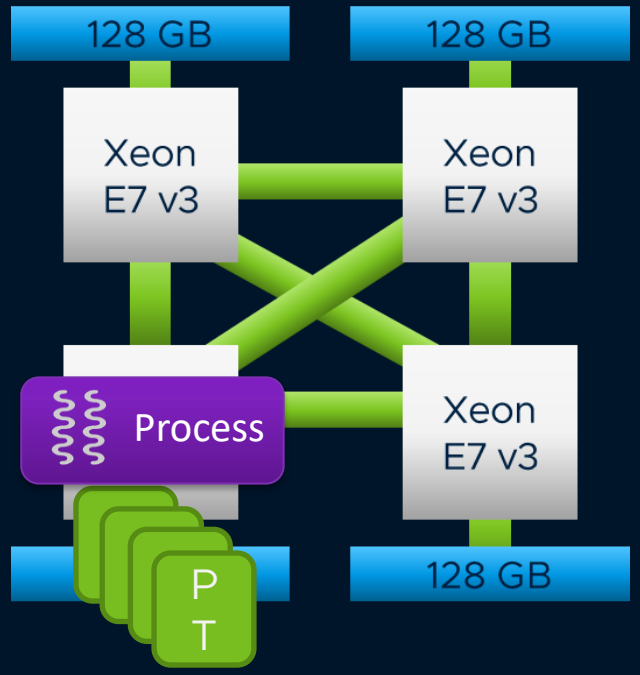
Approach

Multi-Threaded Scenario



Mitosis **replicates** the page table

Workload-Migration Scenario



Mitosis **migrates** the page table

vmware

Mitosis: Keep Page Tables Local using Replication and Migration.

Implementation on Linux / x86_64

Available on GitHub

https://github.com/mitosis-project

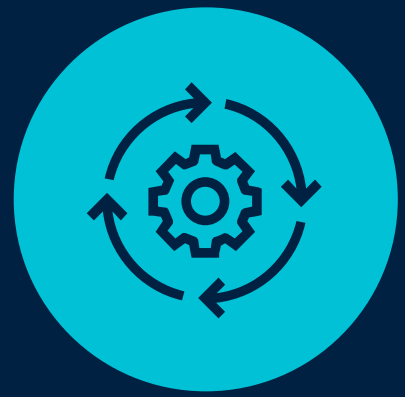
vmware

Mechanism: Replication of Page Tables

Replica Allocation

Consistent updates

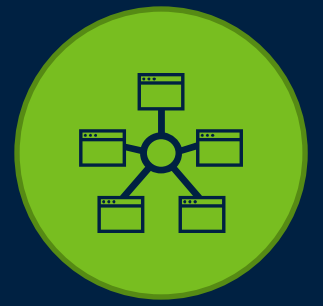
(Correctness)



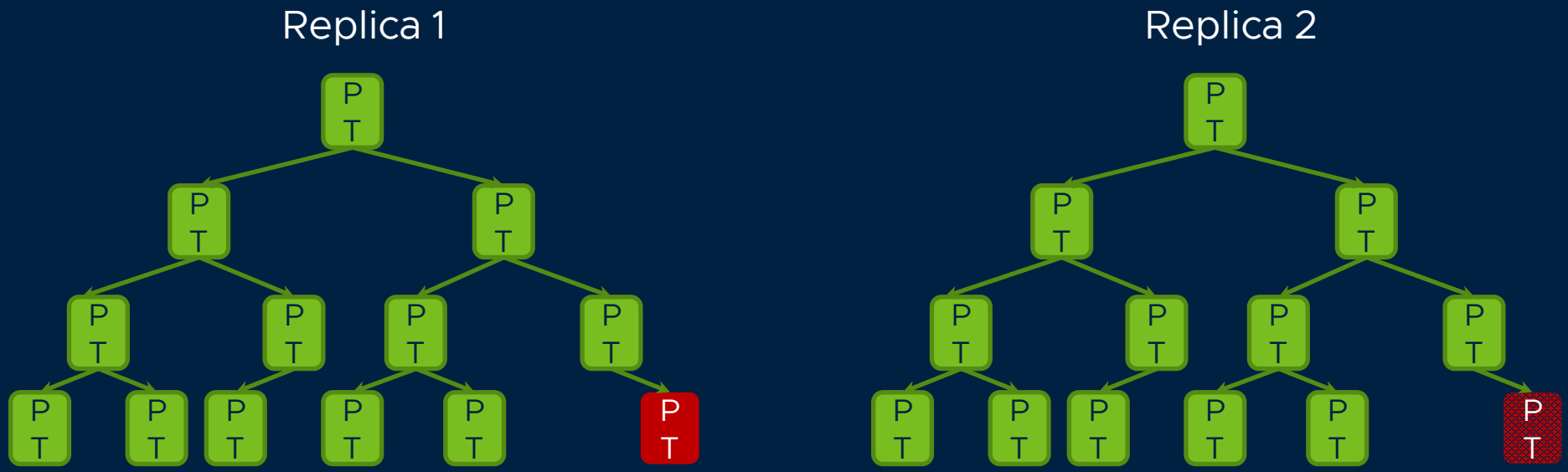
Replica Utilization

(Performance)

Allocation of Page-Table Replicas



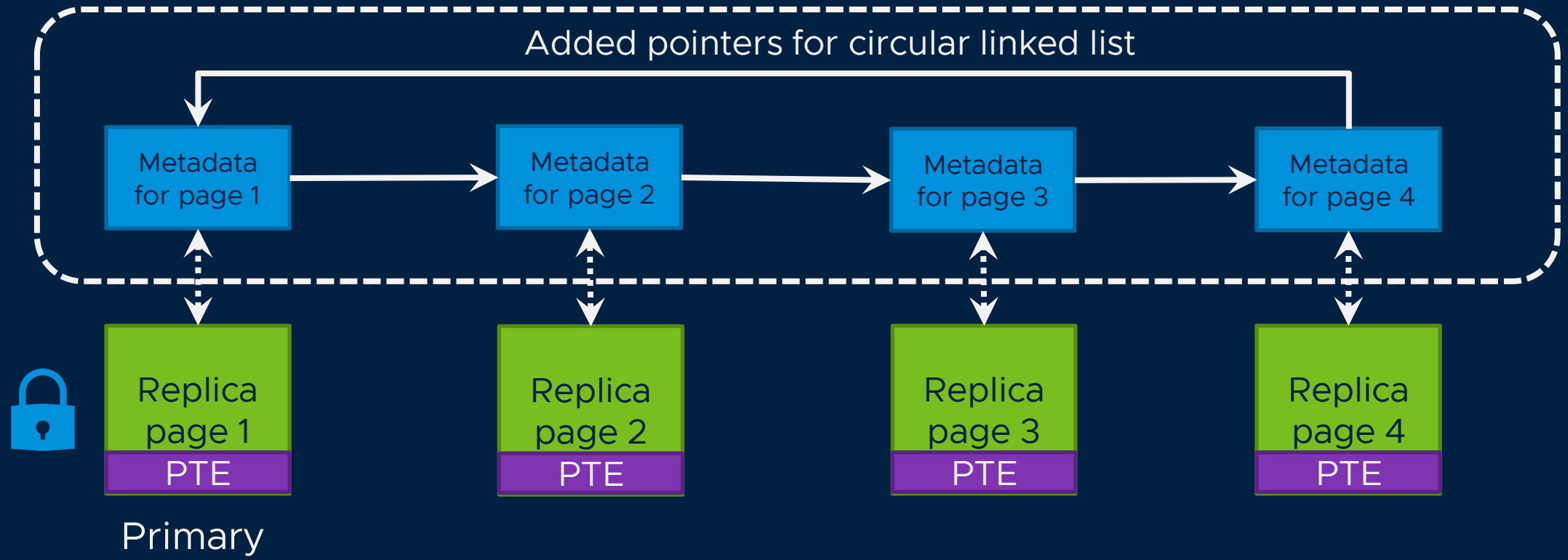
Mitosis eagerly allocates memory for each page table replica.



Consistent Updates to Page-Table Replicas

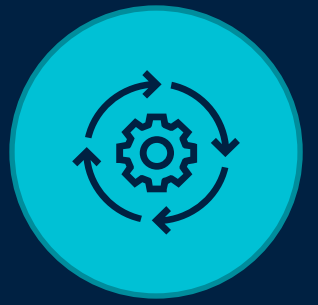
Correctness

Mitosis eagerly keeps replicas consistent on update



Utilizing Page Table Replicas

Performance



Scheduler picks thread to run

Context switch

Select local replica

Write translation base register

Extend Linux' mm_struct
with an array of page table roots
Indexed by NUMA node

```
mm->root[local_node()]
```

PV-Ops: Paravirtualization Operations in the Linux Kernel

PV-Ops Interface

- Set page table entry
- Allocate and free page table page.
- Reading / writing translation base register

native bare metal environment
set_pte() → native_set_pte()

paravirtualized environments
set_pte() → xen_set_pte()

Mitosis PV-Ops Backend

Mitosis implemented as PV-Ops backend.

Extends the native functions to propagate updates to replicas

Adding functions to read entries
→ access & dirty bits

Consistency: Dealing with Access and Dirty Bits

Set by hardware and read by the OS

5 (A)	Accessed; indicates whether software has accessed the 4-KByte page referenced by this entry (see Section 4.8)
6 (D)	Dirty; indicates whether software has written to the 4-KByte page referenced by this entry (see Section 4.8)

Bits set by hardware only in local replica → Inconsistency!

Correctness Problem:
OS relies on those bits in paging / caching decisions

```
if (pte_young(*ptep) || pte_dirty(*ptep)) {
   // do something
}
```

Solution: OR-ing entries of all replicas to determine dirty / accessed state!

Mitosis Policy Settings

System Wide Policy

procfs entries to globally enable/disable Mitosis

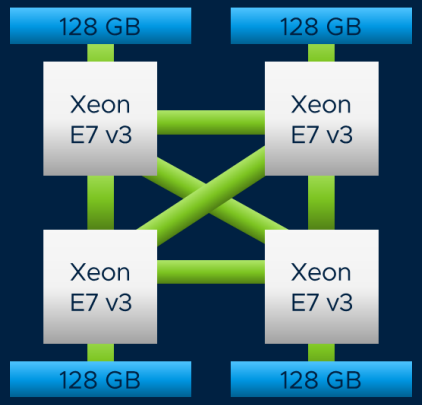
/proc/sys/kernel/pgtable_replication

Per Application Policy

Integration in numactl / libnuma

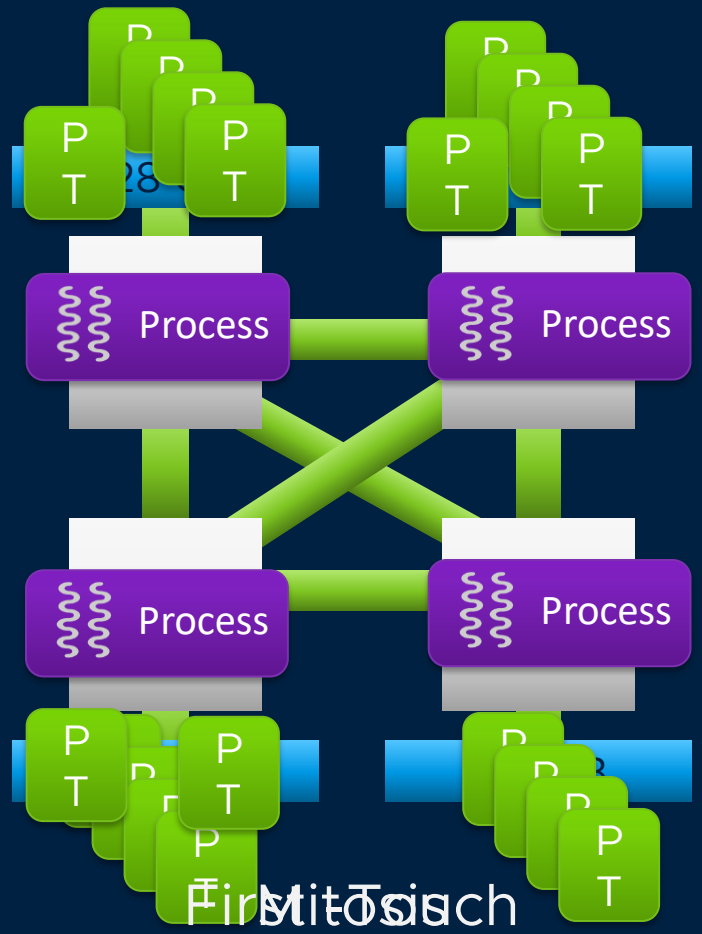
numactl –r 0-3 <workload>

Results

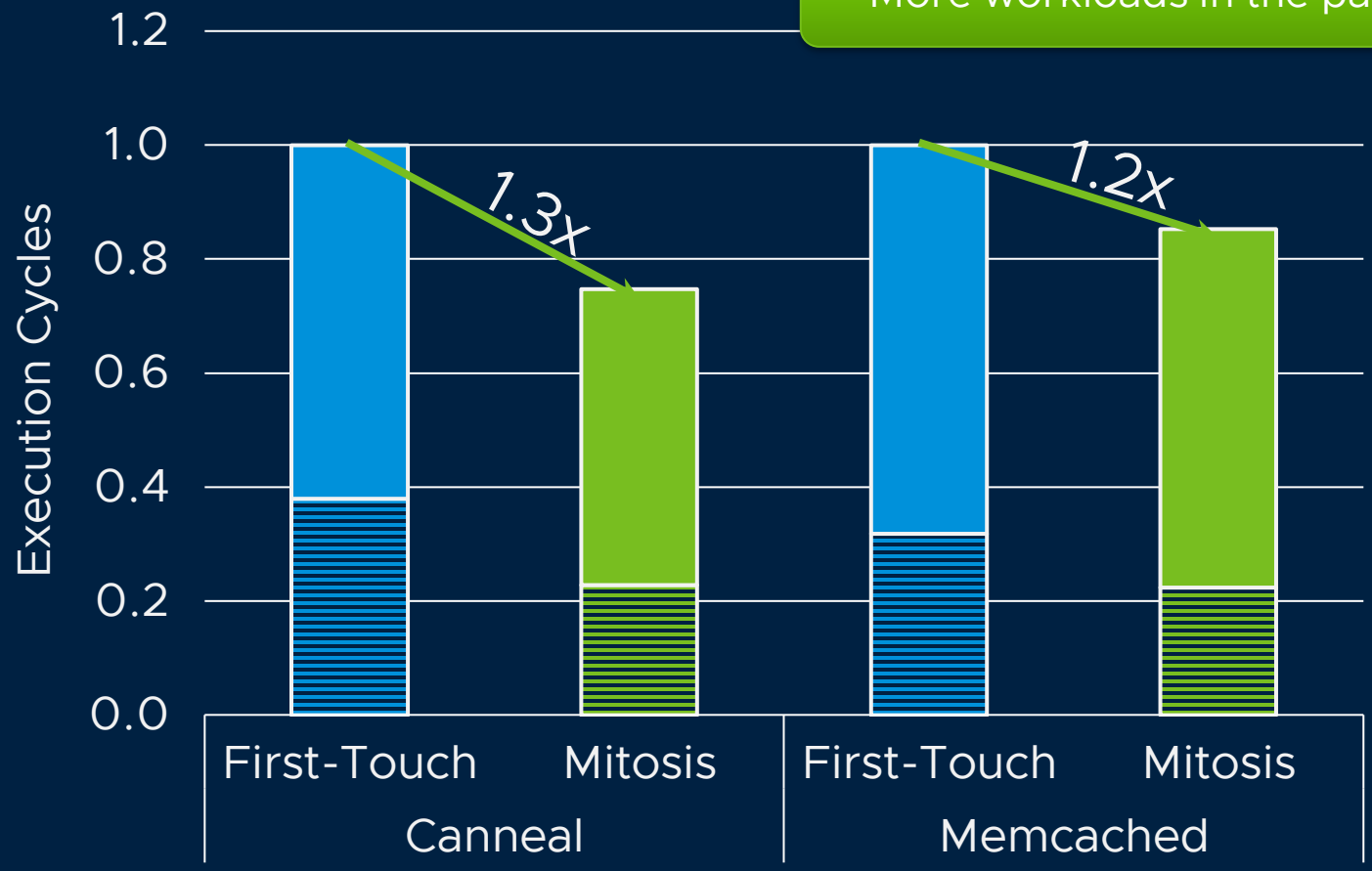


Measured on
4x12 Intel Xeon E7-4850 v3,
with 512GB RAM (4x128GB)

Evaluation: Multi-Threaded Scenario with Mitosis



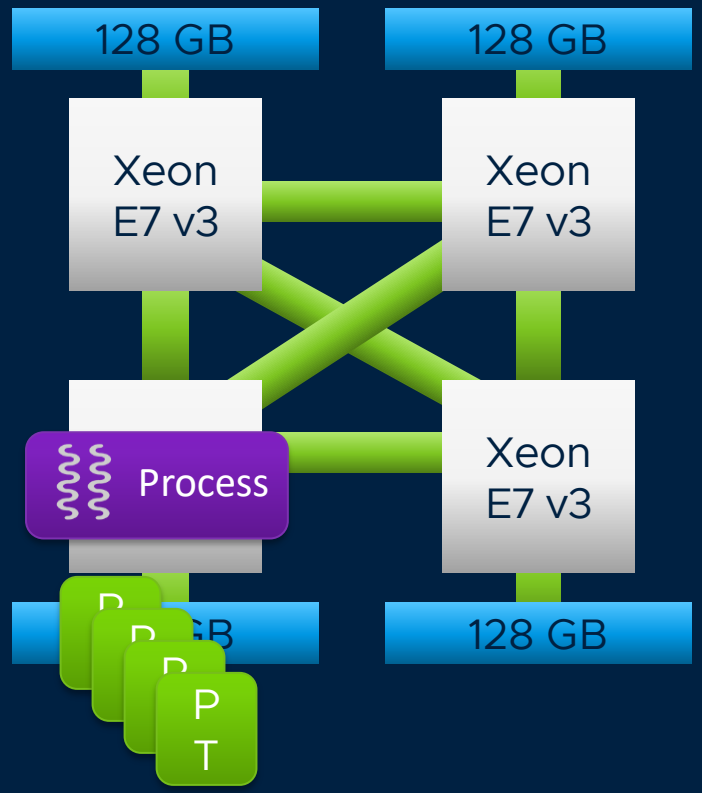
More workloads in the paper

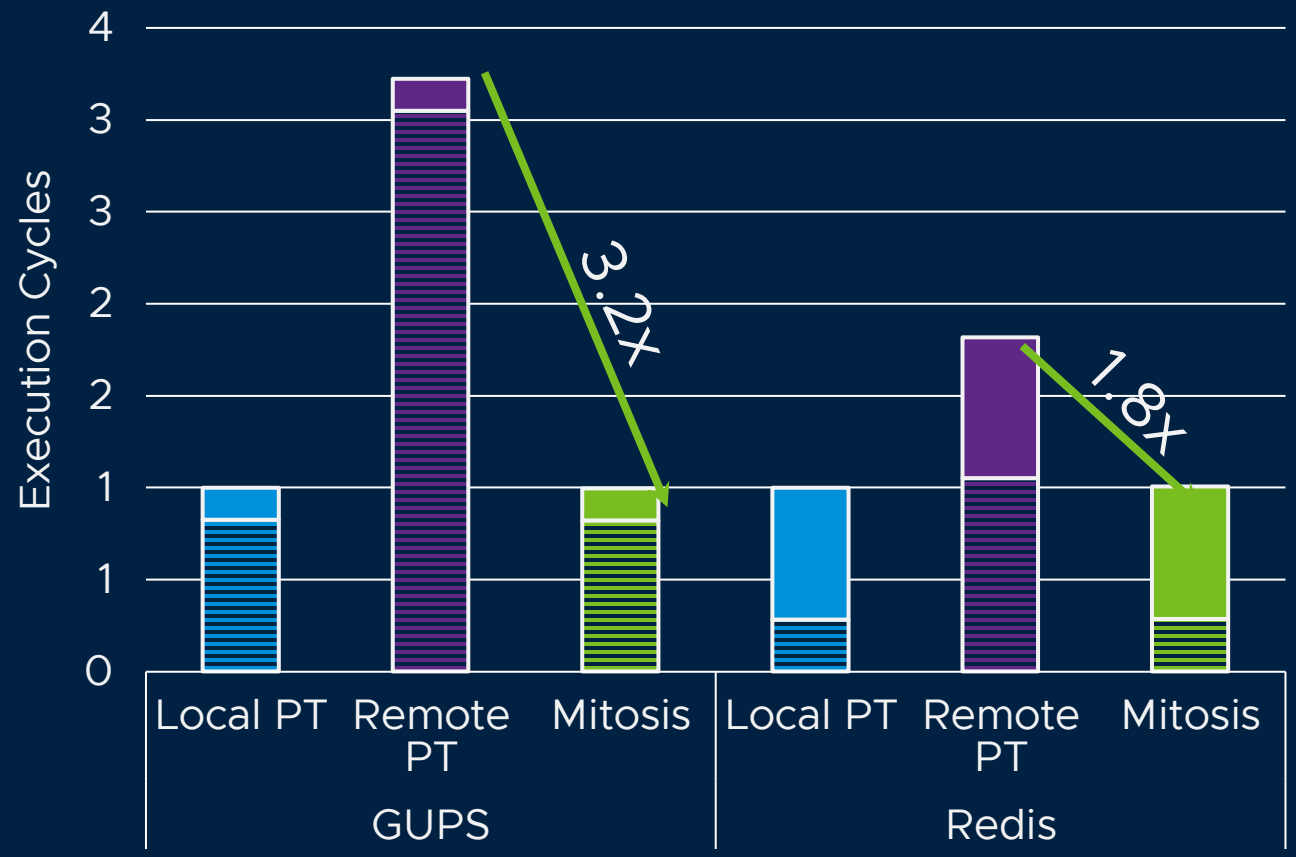


Performance improvement with replication of page tables

Evaluation: Workload-Migration Scenario with Mitosis

More workloads in the paper





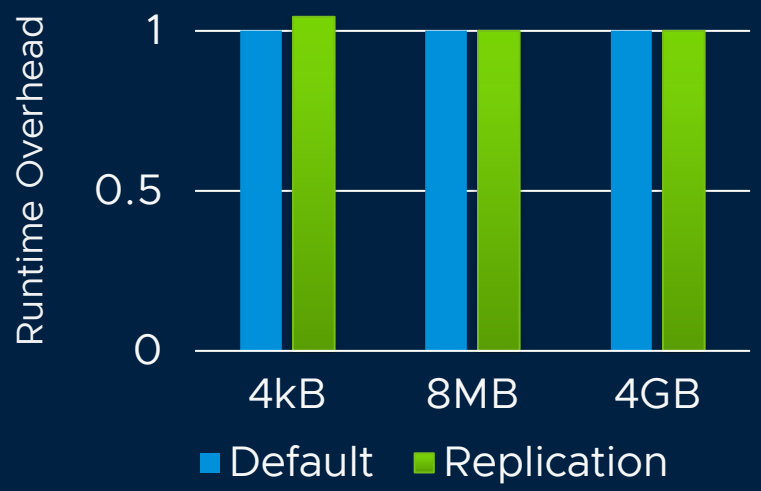
Performance improvement with migration of page tables

vmware

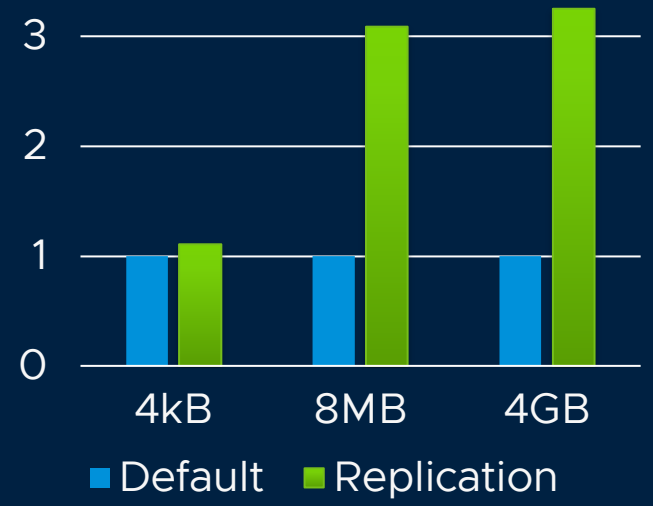
Replica Maintenance Overhead

Mitosis with 4-Way Replication

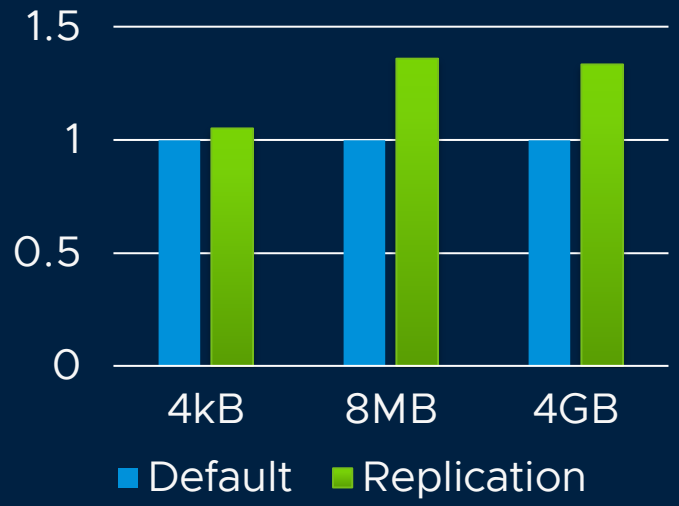
mmap()



mprotect()



munmap()



Space Overhead: 0.6% of additional memory for 4-way replication

vmware®

Mitosis Summary

Click to edit optional subtitle

Available on GitHub

https://github.com/mitosis-project

Mitosis mitigates NUMA effects on page table walks

NUMA effects

The first study to show NUMA effects on page table walks.

Mitosis Linux Implementation

Page table replication as PV-Ops backend.

Integration into libnuma and numactl.

Results

3.2x speed up for workload migration scenario

Up to 1.3x speed up for multi-socket scenario .

vmware®