FAST LOCAL PAGE TABLES FOR NUMA SERVERS WITH MITOSIS

Reto Achermann, University of British Columbia

ICSA COLLOQUIUM – UNIVERSITY OF EDINBURGH, JUNE 8, 2021

MITOSIS: MITIGATING NUMA EFFECTS ON PAGE TABLE WALKS

Up to 3.2x speed up for single threaded applications.

Up to 1.6x speed up on multi-threaded applications

No modifications of the application workloads

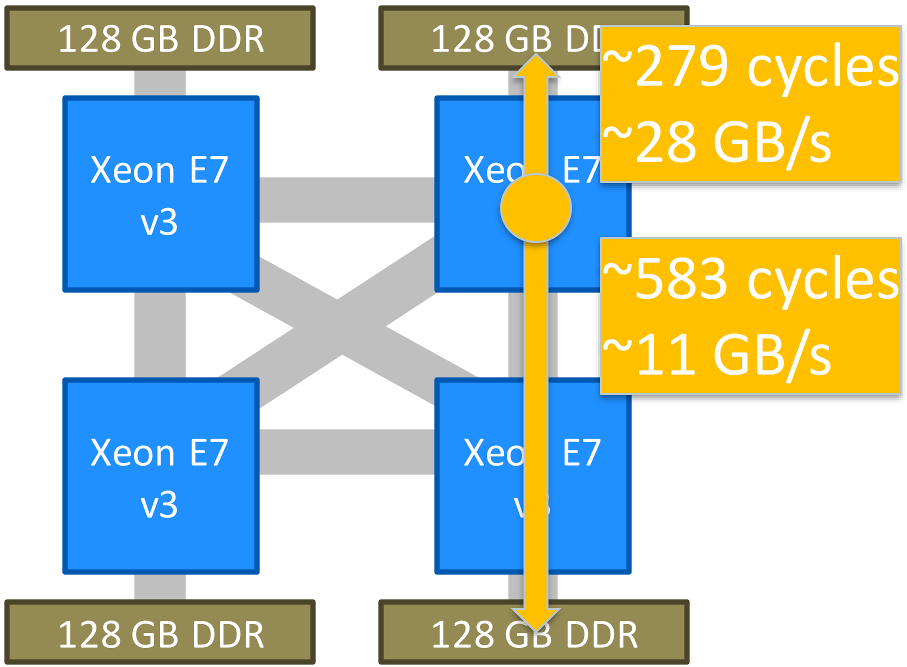
Linux (native) and KVM-based virtual machines

NUMA EFFECTS ON LARGE MULTI-SOCKET MACHINES

Bandwidth & capacity limited per processor socket

More bandwidth & capacity needed
→ Multi-socket machine

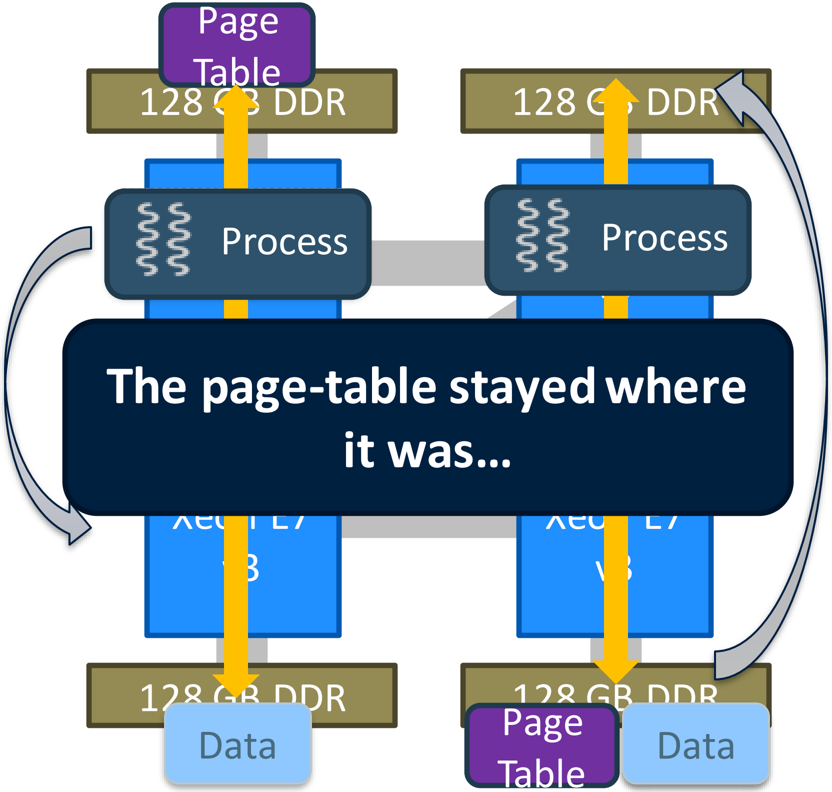
Non-uniform memory access (NUMA) machine
→ Memory accesses experience different performance characteristics



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

KEEPING PROCESS THREADS AND DATA CLOSE

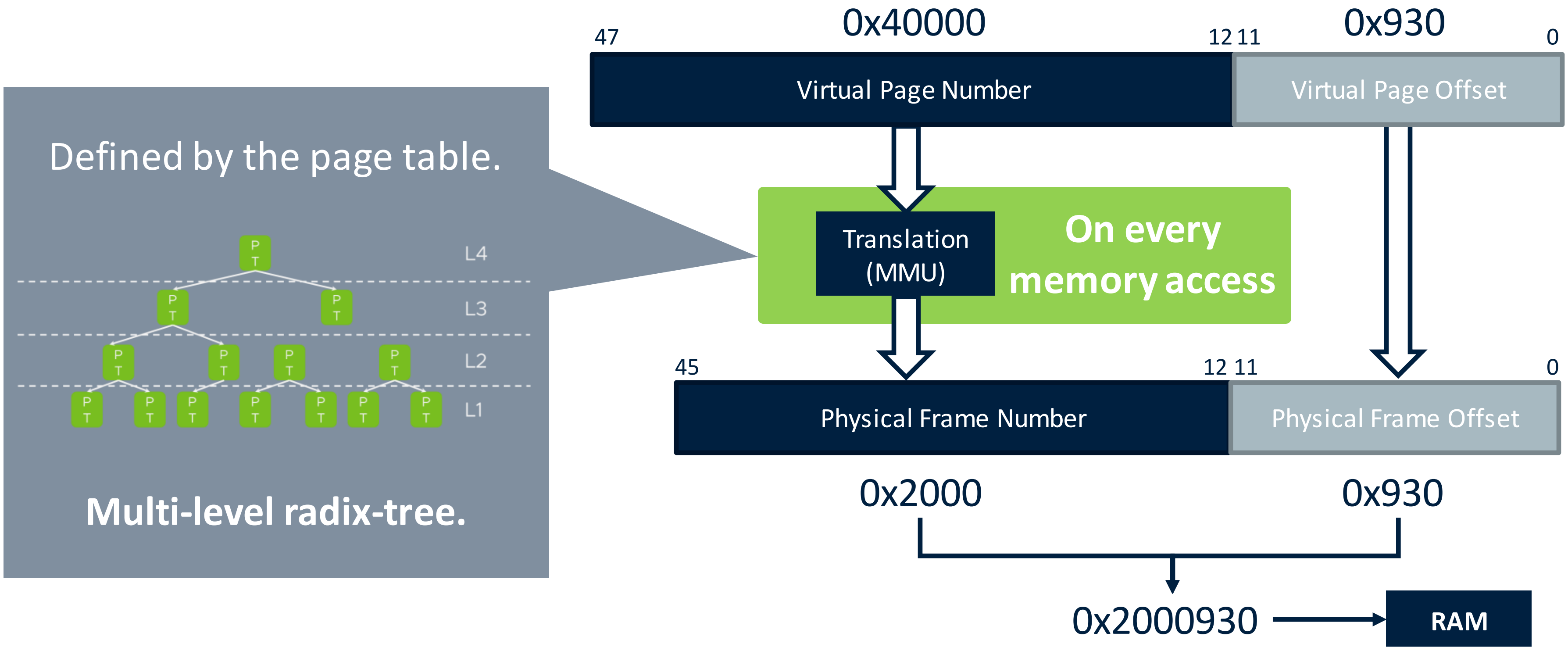
NUMA-aware scheduling



NUMA-aware data migration

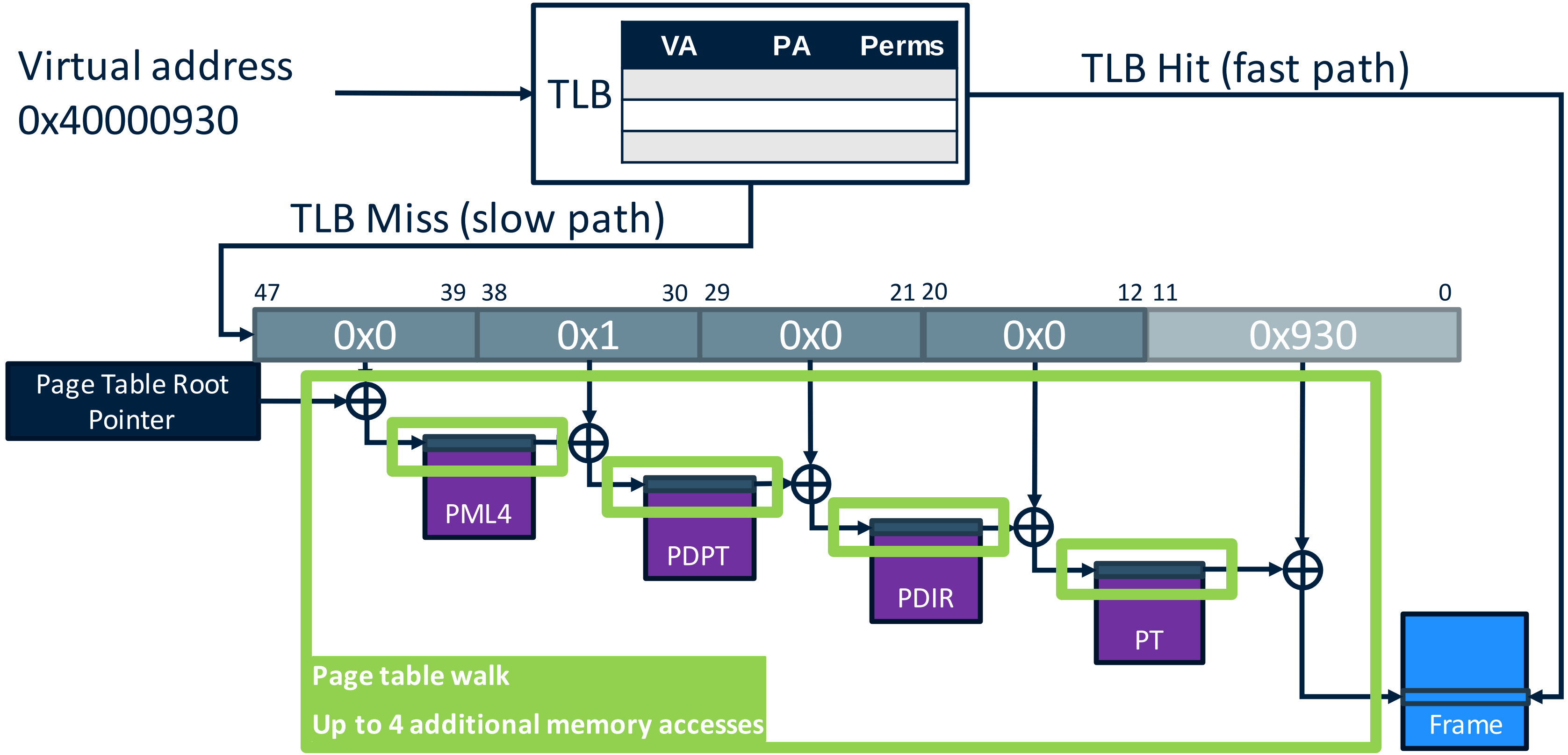
RECAP: VIRTUAL TO PHYSICAL TRANSLATION

Processes deal with virtual addresses



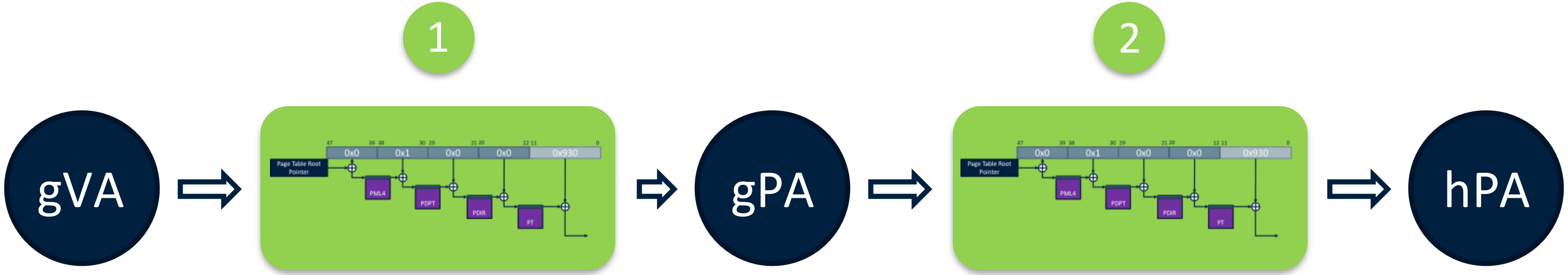
memory knows physical addresses

TLB MISSES TRIGGER PAGE TABLE WALKS



TWO STAGE TRANSLATIONS OF VIRTUAL MACHINES

Modern processors support two-stage translation schemes



guest virtual address

guest page-table (gPT)

guest physical address

extended page-table (ePT)

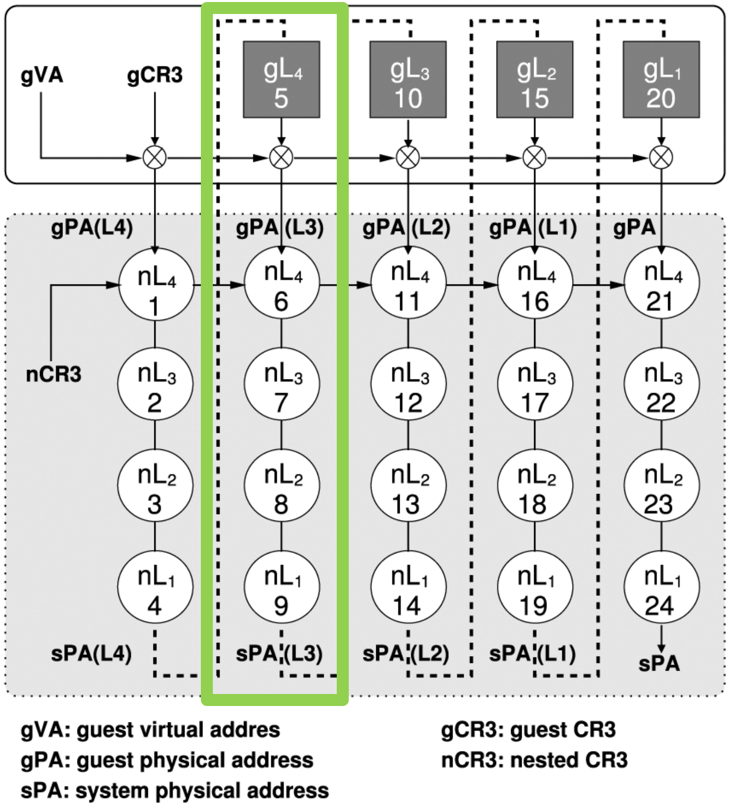
host physical address

TWO STAGE TRANSLATIONS OF VIRTUAL MACHINES

Translation is **not** sequential!

Page table walk for gVA -> gPA translation uses guest physical addresses.
→ guest page table walk needs translation

Up to 24 memory accesses to perform gVA -> hPA translation



J. Ahn, S. Jin and J. Huh, "Fast Two-Level Address Translation for Virtualized Systems"

THE PROBLEM IN SUMMARY

1. More DRAM capacity & stagnating TLB size
   → decreasing TLB coverage and more frequent TLB misses.

2. These TLB misses require multiple memory accesses

3. Virtualization: two sets of page tables and nested page table walks.

4. Contention may evict page tables from caches
   → page walker accesses end up in DRAM

5. Accessing DRAM may be subject to NUMA effects

NUMA effects on page table walks – a systematic analysis.

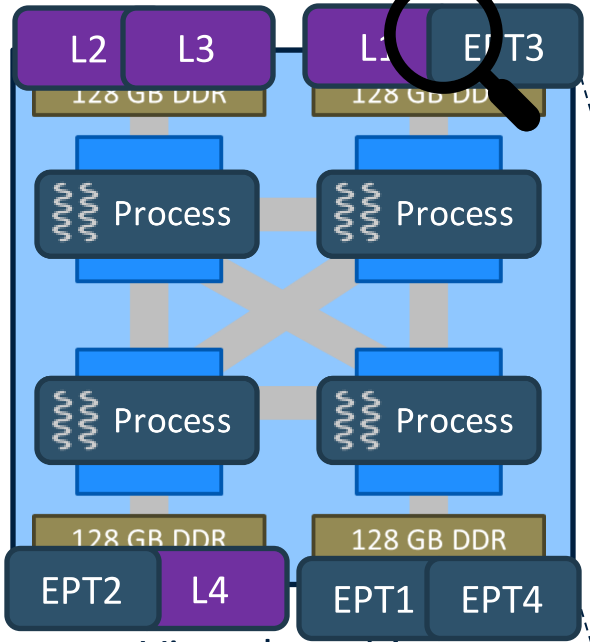
REMOTE PAGE TABLE WALKS ON NUMA SYSTEMS

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

Systematic analysis with different scenarios:

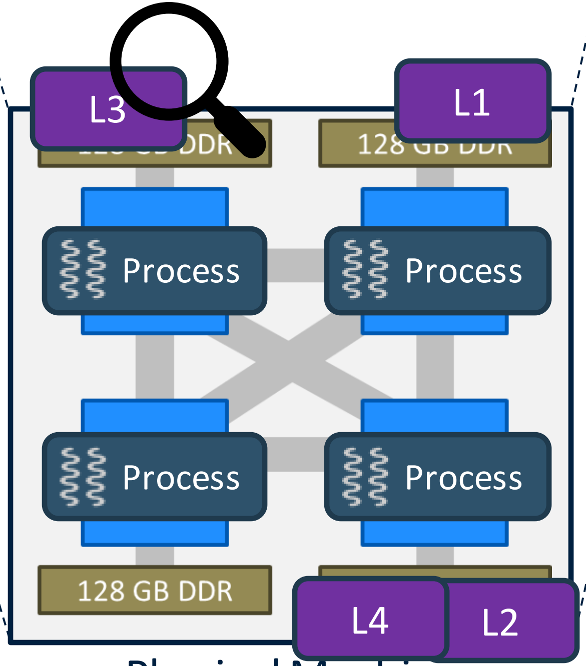
Multi-Threaded Scenario	Workload-Migration Scenario

MANY THREADS – ONE PAGE TABLE

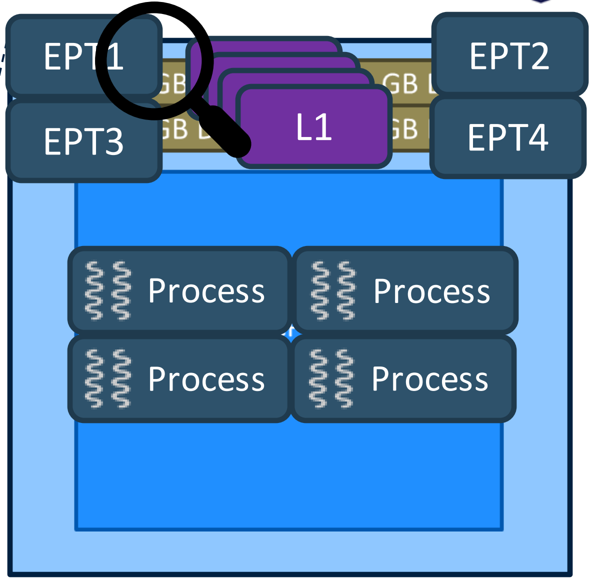


Virtual machine
NUMA visible
configuration

Capture the page table
Analyze NUMA affinity
of page table entries



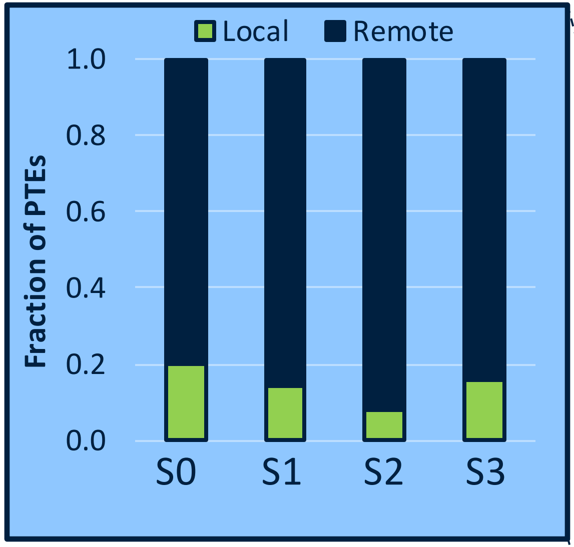
Physical Machine



Virtual machine
NUMA oblivious
configuration

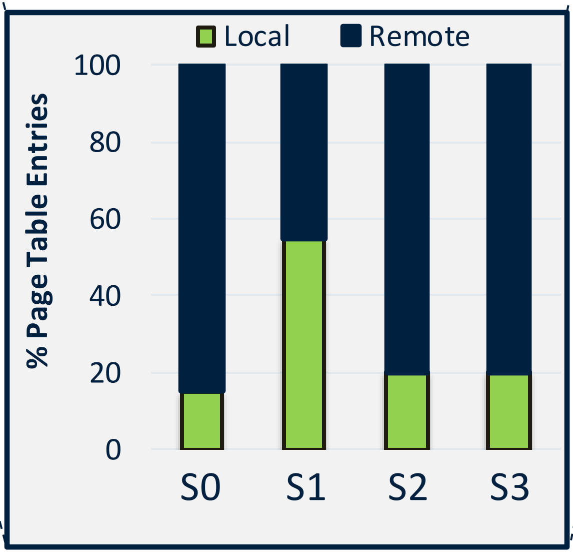
Multi-Threaded Scenario

WHERE DO PAGE TABLE ENTRIES POINT TO?

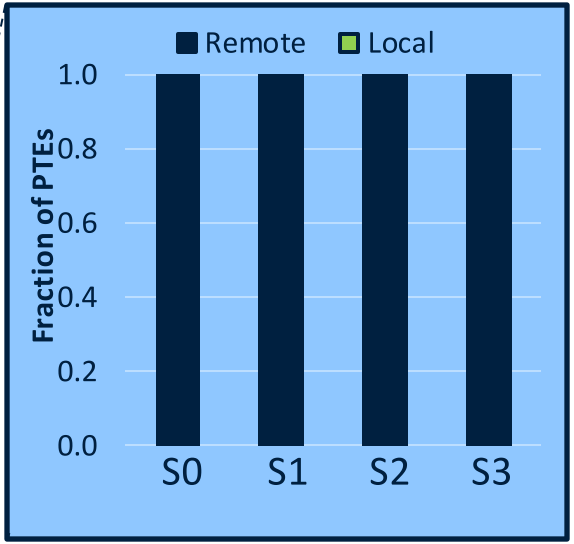


Virtual machine
NUMA visible
configuration

Majority of TLB misses require remote memory accesses.



Physical Machine



Virtual machine
NUMA oblivious
configuration

Workload: Graph500

REMOTE PAGE TABLE WALKS ON NUMA SYSTEMS

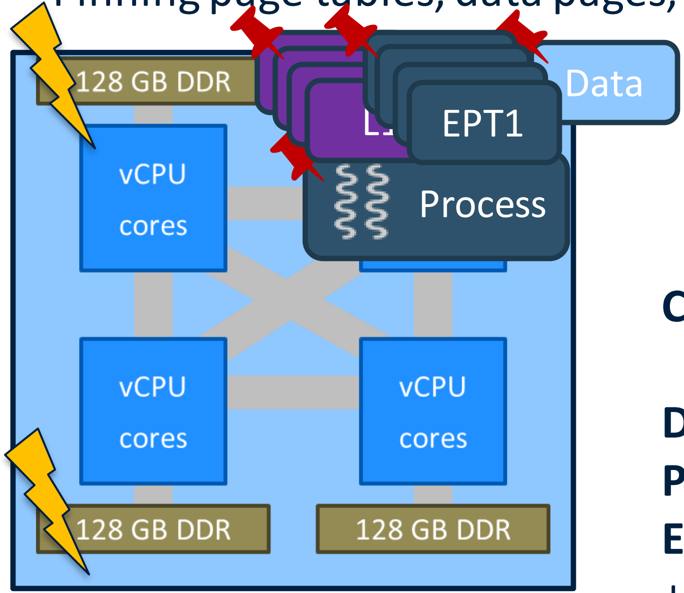
How often is the page table remote and what's the resulting slowdown?

Systematic analysis with different scenarios:

Multi-Threaded Scenario	Workload-Migration Scenario
Majority of TLB misses require remote memory accesses	

REMOTE PAGE TABLE WALKS ON NUMA SYSTEMS

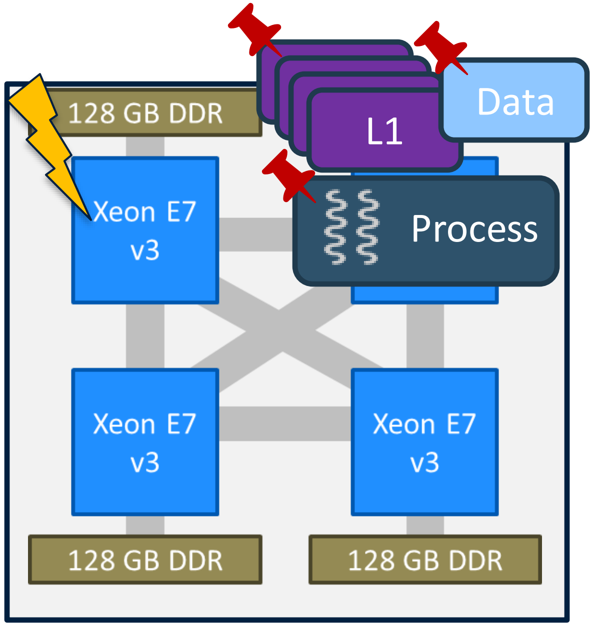
Pinning page tables, data pages, threads to specific cores



Virtual machine
NUMA visible
configuration

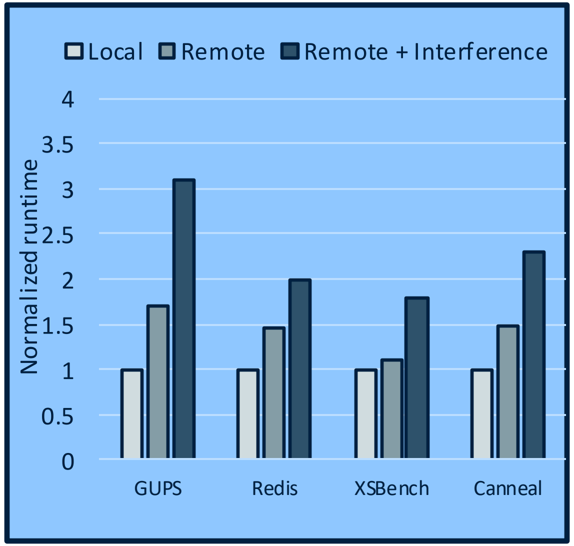
Configurations:

Data: Local/Remote
Page Table: Local/Remote
EPT: Local/Remote
+ **Interference** on PT node



Physical Machine

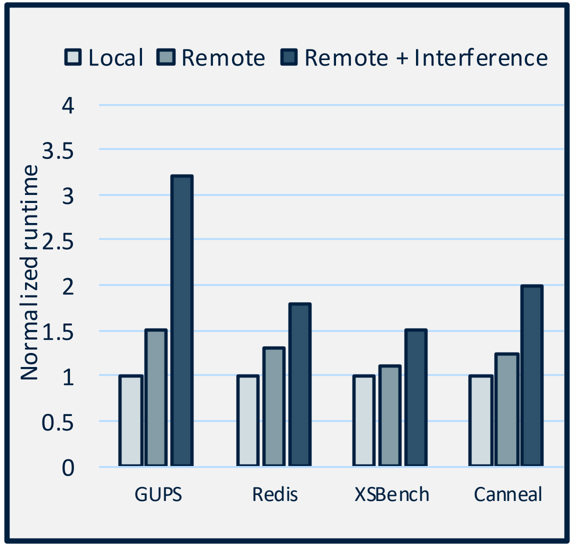
REMOTE PAGE TABLE WALKS ON NUMA SYSTEMS



Virtual machine **NUMA visible** configuration

Remote page walks during TLB misses have significant impact on application performance

More configurations and workloads in the paper



Physical Machine

SUMMARY: MEMORY ACCESSES FROM PAGE WALKER EXPERIENCE NUMA EFFECTS

TLB misses result in **additional** memory accesses

Systematic analysis with different scenarios:

Multi-Threaded Scenario	Workload-Migration Scenario
Majority of TLB misses require remote memory accesses	Remote page walks during TLB misses have significant impact on application performance

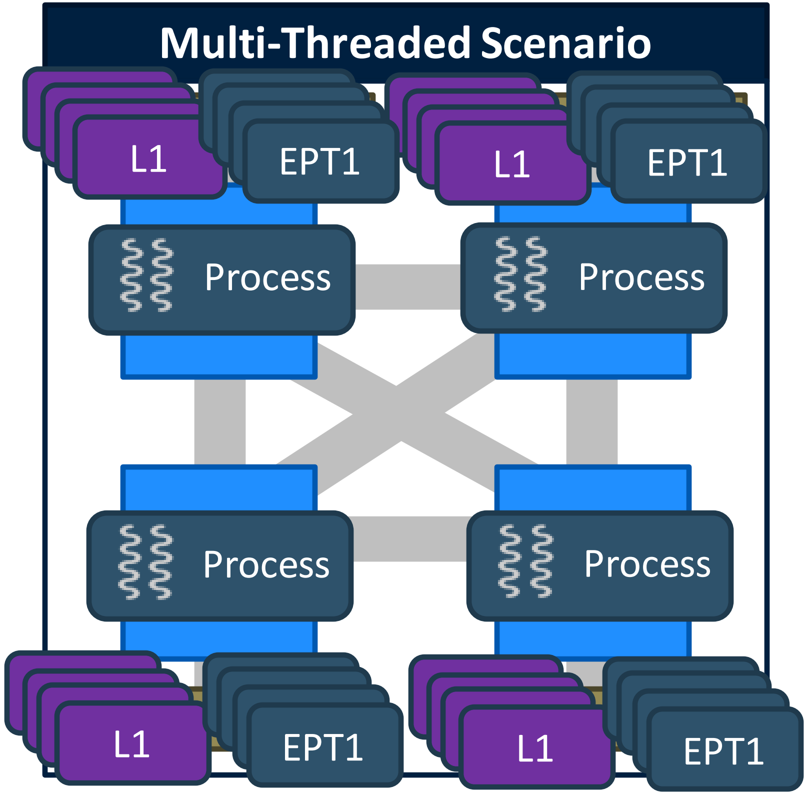
Mitosis: Keep Page Tables Local using Replication and Migration.

Design and Implementation on Linux/KVM / x86_64

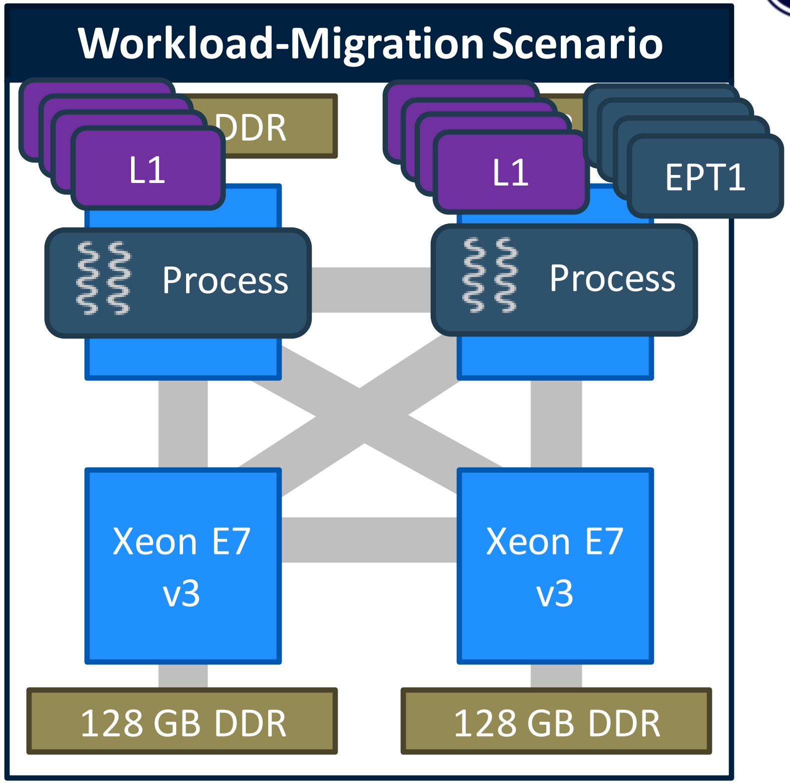
Available on GitHub

https://github.com/mitosis-project

MITOSIS+VMITOSIS: MAKING PAGE TABLES LOCAL

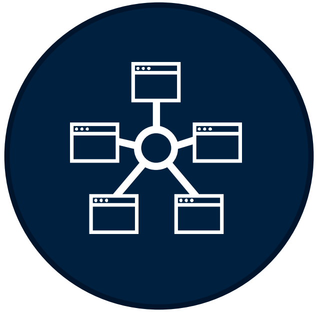


Mitosis **replicates** the page table



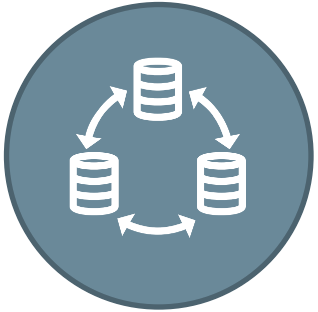
Mitosis **migrates** the page table

REPLICATION OF PAGE TABLES



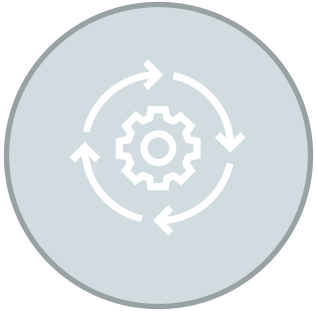
Allocation

(Performance)



Consistency

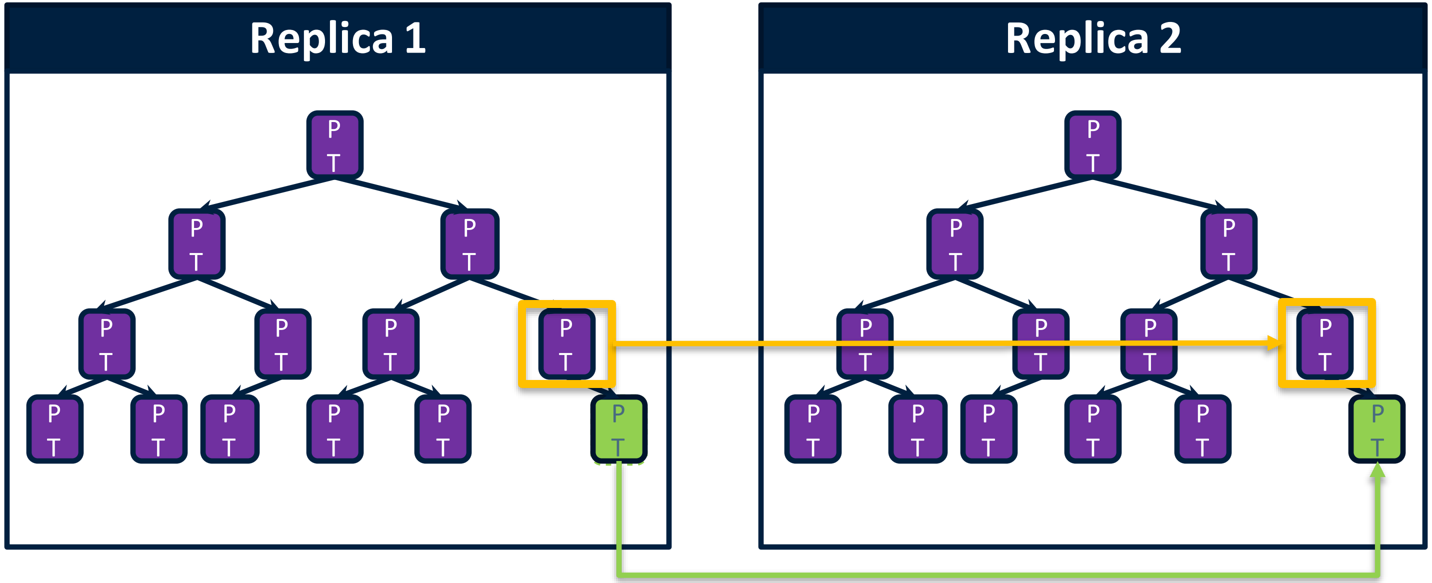
(Correctness)



Utilization

(Performance)

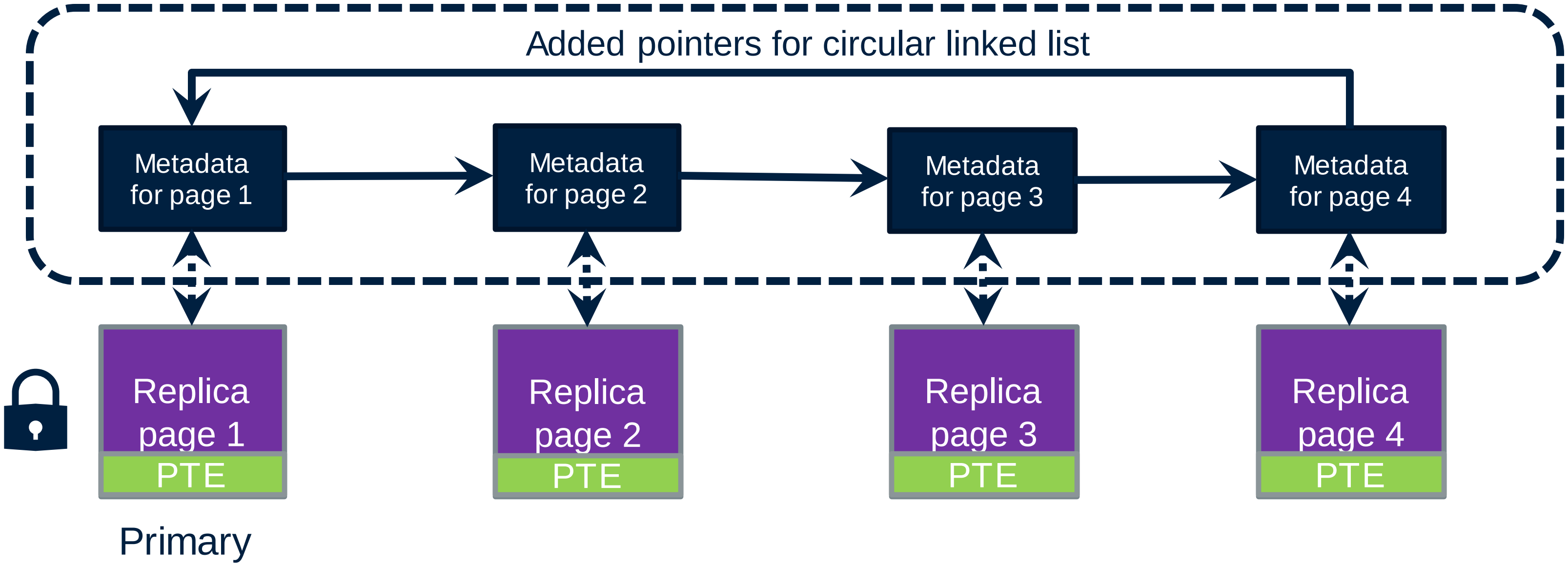
MITOSIS EAGERLY ALLOCATES REPLICA PAGE TABLES



We need to efficiently find the new replica's parent page table.

allocate all replica page tables from the corresponding NUMA node

EFFICIENT PROPAGATION OF UPDATES



SELECTING THE RIGHT REPLICA

Scheduler picks thread to run

↓

Context switch

↓

Select local replica

↓

Write translation base register

Extend Linux' mm_struct

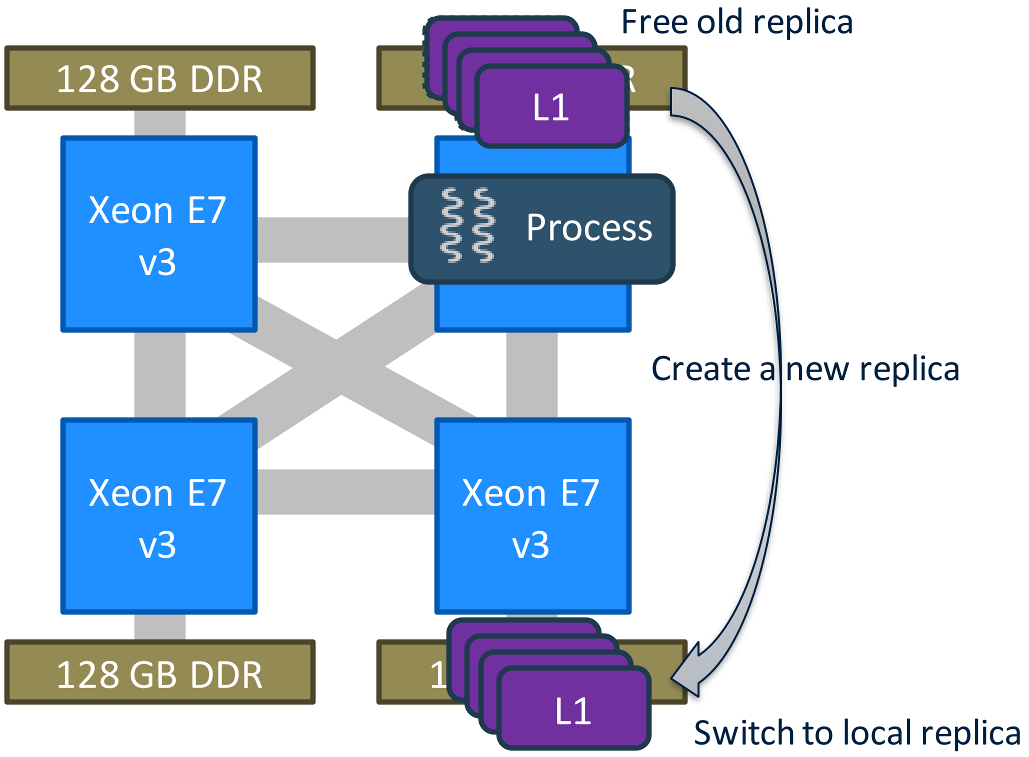
with an array of page table roots

Indexed by NUMA node

mm->root[local_node()]

(KVM similar)

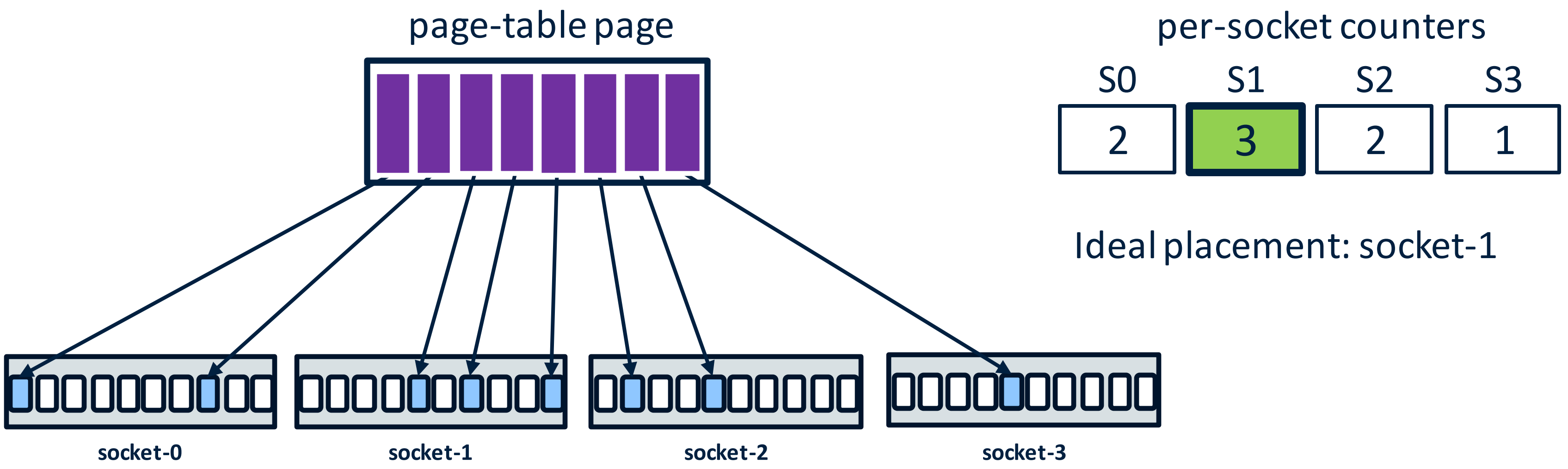
MIGRATING PAGE TABLES: REPLICATE + FREE



Linux already migrates data pages, can we hook in there?

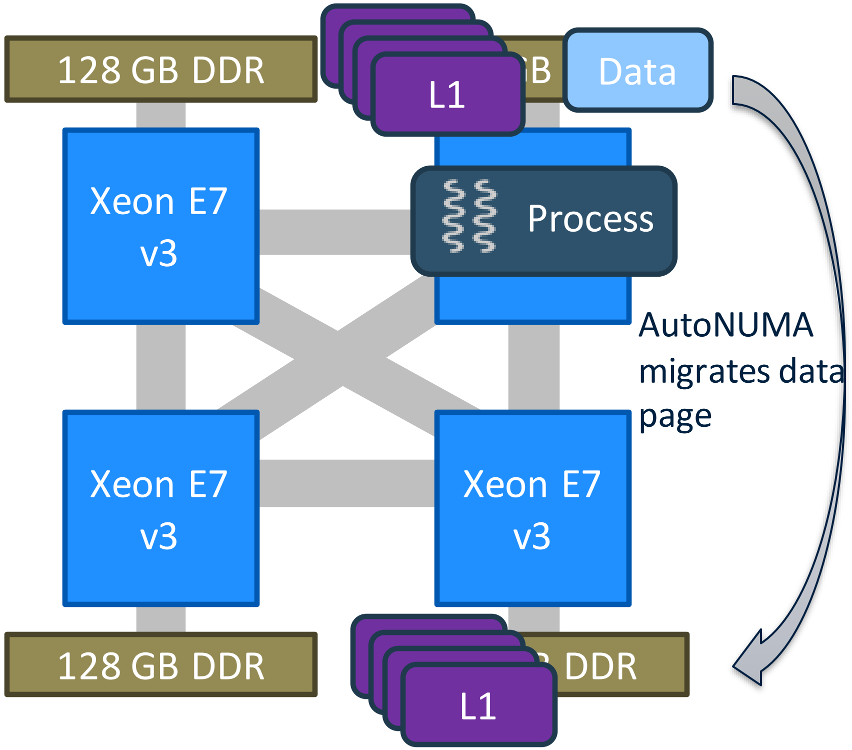
MIGRATION OF PAGE TABLES

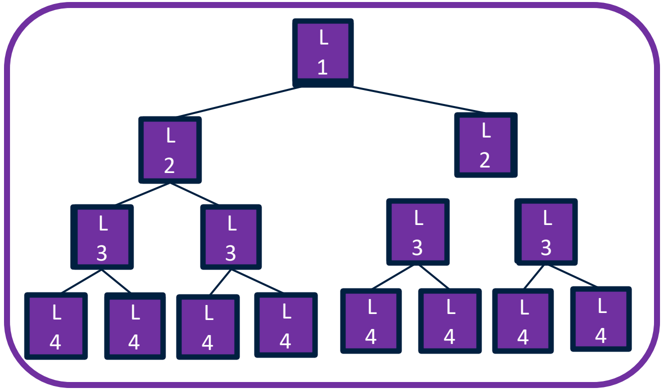
Locality Invariant: Placing page-table pages with majority of their children

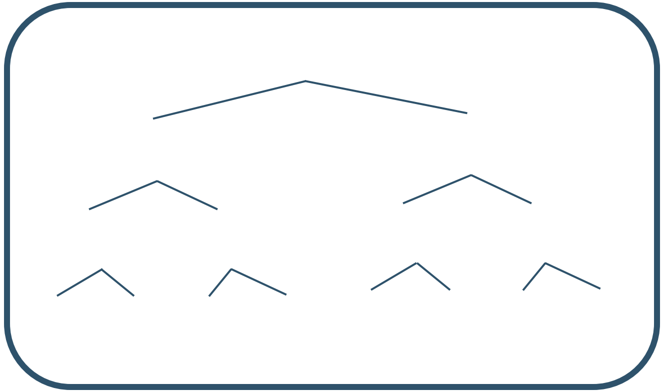


MIGRATION

MECHANISM: AUTONUMA FOR PAGE TABLES







USING MITOSIS

No application modifications

```
numactl -r <sockets> <workload>
```

*) Requires Mitosis enabled kernel and libnuma

Available on GitHub

https://github.com/mitosis-project

REPLICATION WITHOUT NUMA TOPOLOGY INFORMATION

NUMA visible case is easy:

1) get NUMA node of CPU

2) select local replica for gPT and ePT

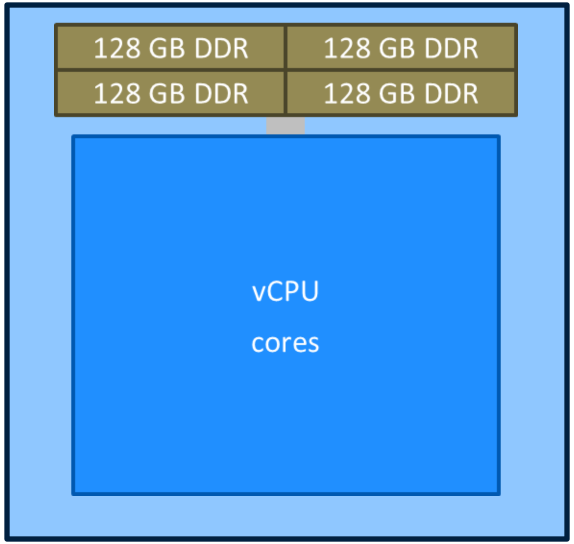
Problem: NUMA topology is not always available

(e.g., hidden by the hypervisor)

Requirements:

vCPU -> NUMA node

memory page -> NUMA node



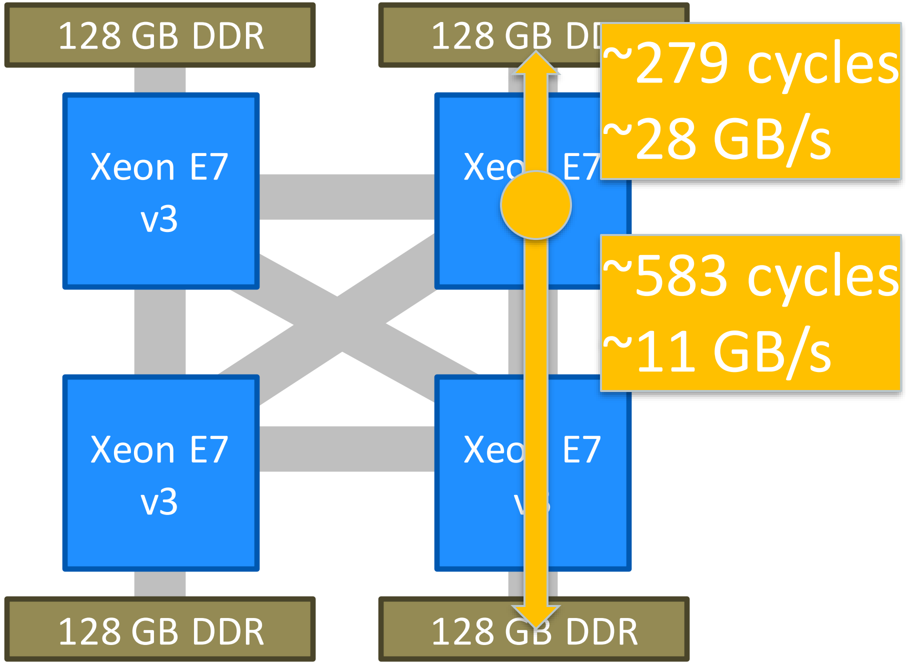
Virtual machine
NUMA oblivious
configuration

RECONSTRUCTING NUMA TOPOLOGY

Measure **cache-line transfer latency** between any pair of cores.

	0	1	2	3	4
0	-	272	590	587	595
1	265	-	593	585	580
2	588	585	-	269	579
3	590	589	271	-	581

vNUMA 0	0, 1, …
vNUMA 1	2, 3, …
vNUMA 2	4, 5, …
vNUMA 3	6, 7, …



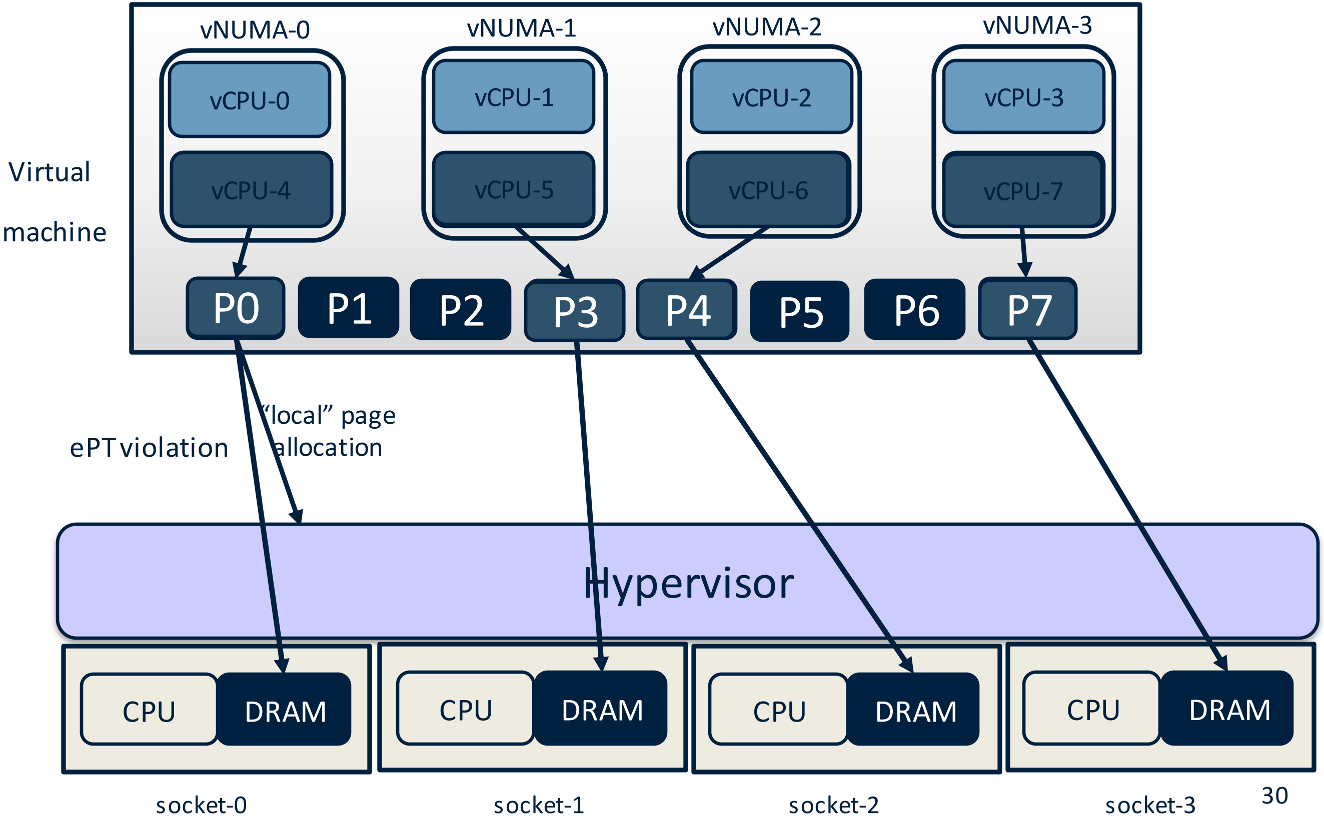
Measured on 4x12 Intel Xeon E7-4850 v3, with 512GB RAM (4x128GB)

ALLOCATING LOCAL PAGES

Exploit the first-touch / local allocation policy

Chose vCPU to access the page.

Hypervisor handles ePT violation, and maps local page



Results

EVALUATION

Multi-Threaded Scenario

What is the resulting speedup on multi-threaded workloads with Mitosis?

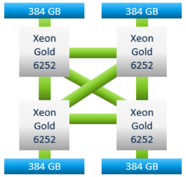
Workload-Migration Scenario

Can Mitosis prevent the significant slowdown?

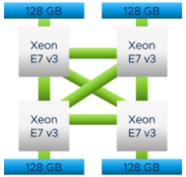
Mitosis Overheads

What is the overhead of Mitosis?

Linux and Linux+KVM

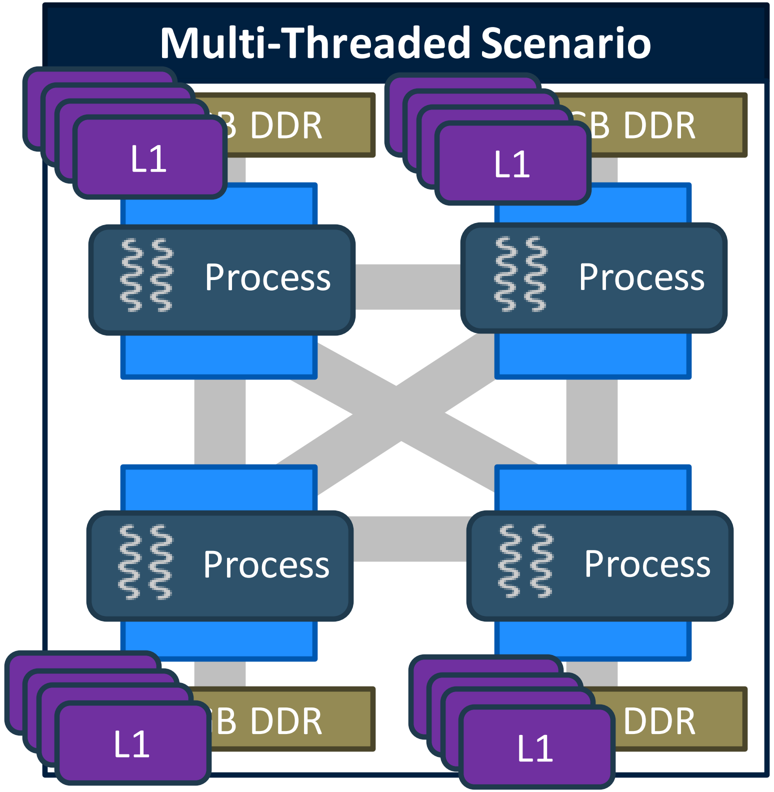


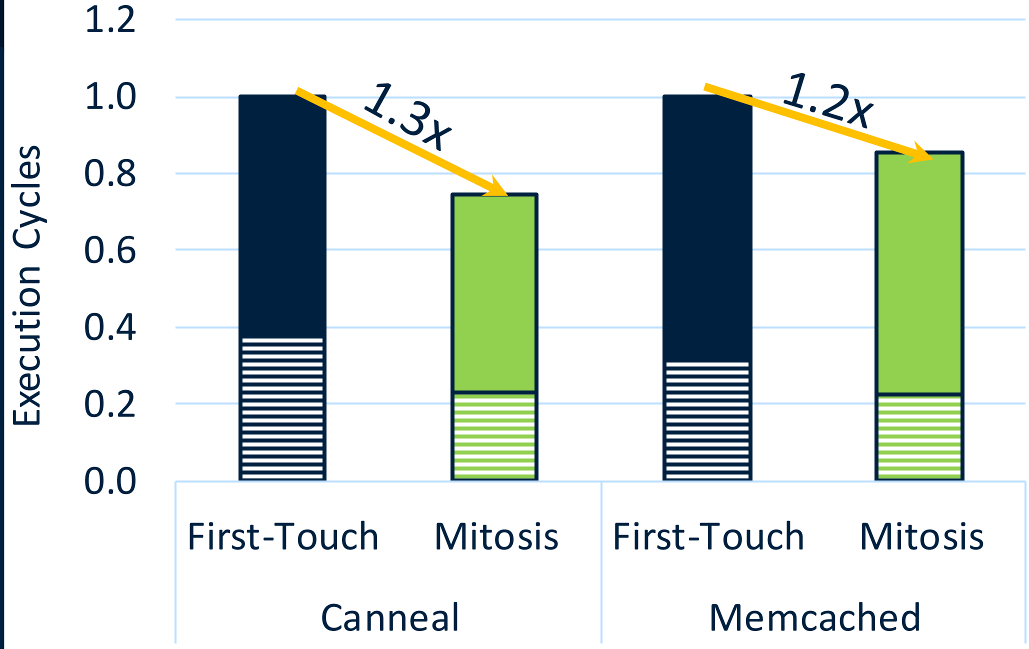
4x26 Intel Xeon Gold 6252 with 1.5TiB RAM (4x384 GiB)



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

MITOSIS ON NATIVE LINUX

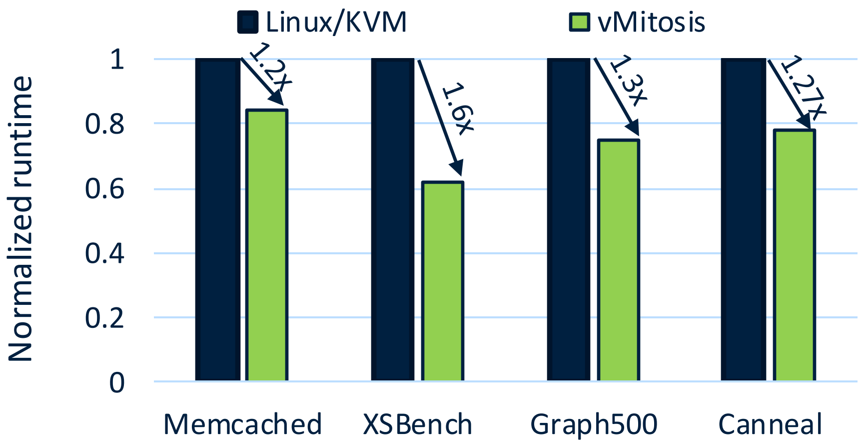




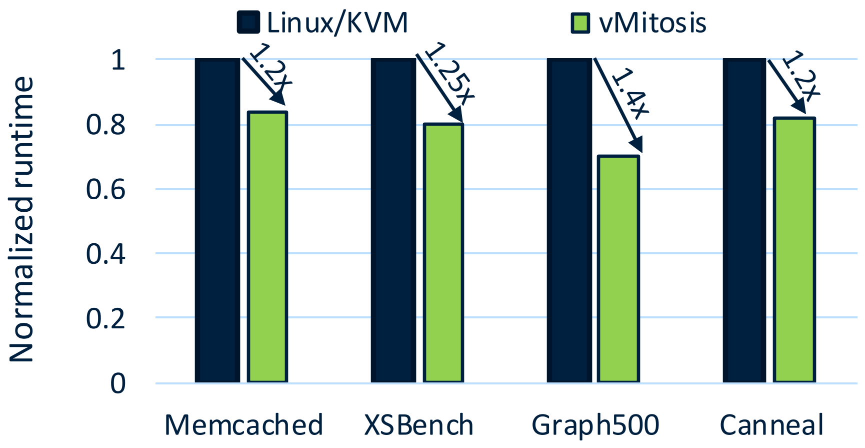
Mitosis **replicates** the page table

MITOSIS ON KVM VIRTUAL MACHINE

NUMA VISIBLE

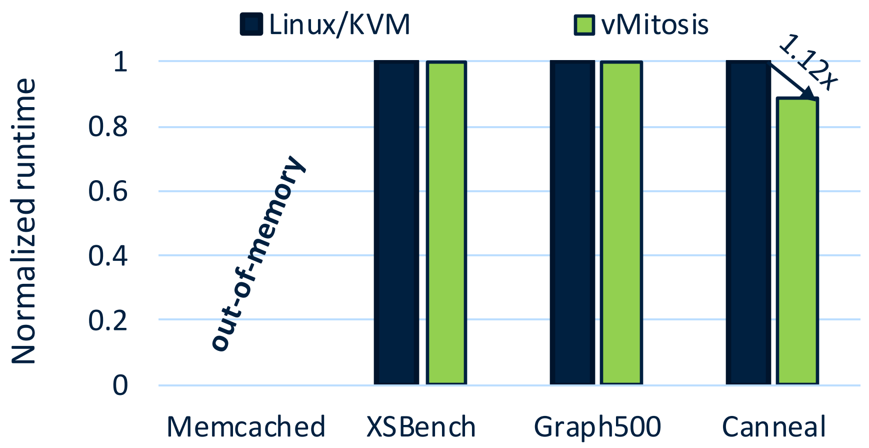


NUMA OBLIVIOUS

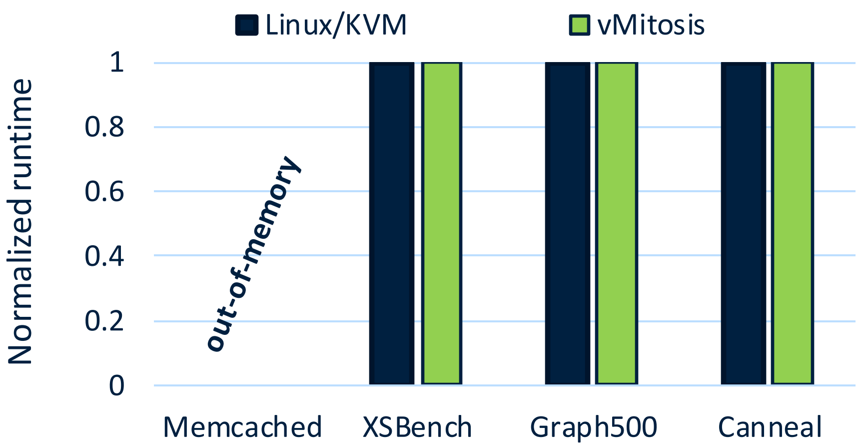


MITOSIS WITH TRANSPARENT HUGE PAGES (THP)

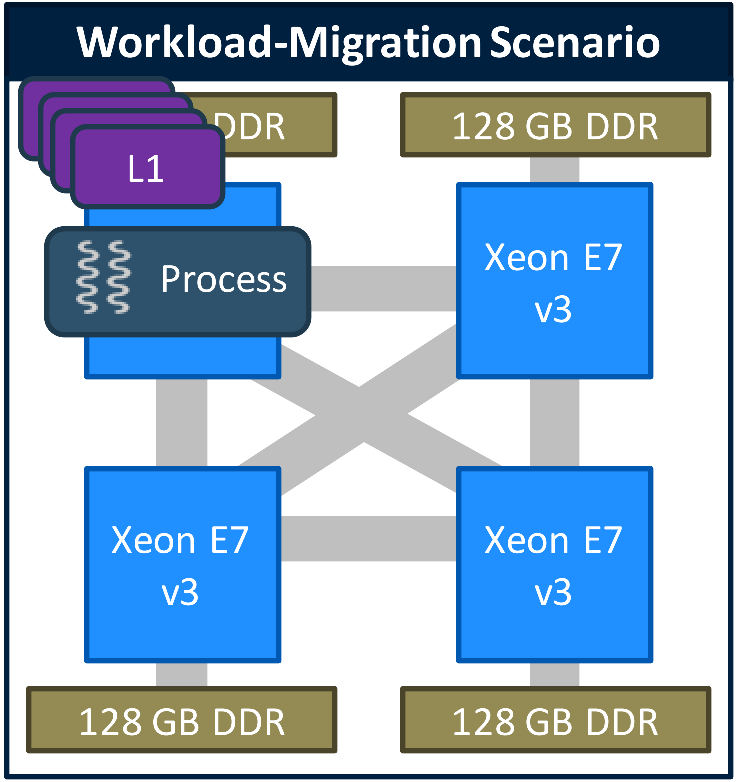
NUMA VISIBLE

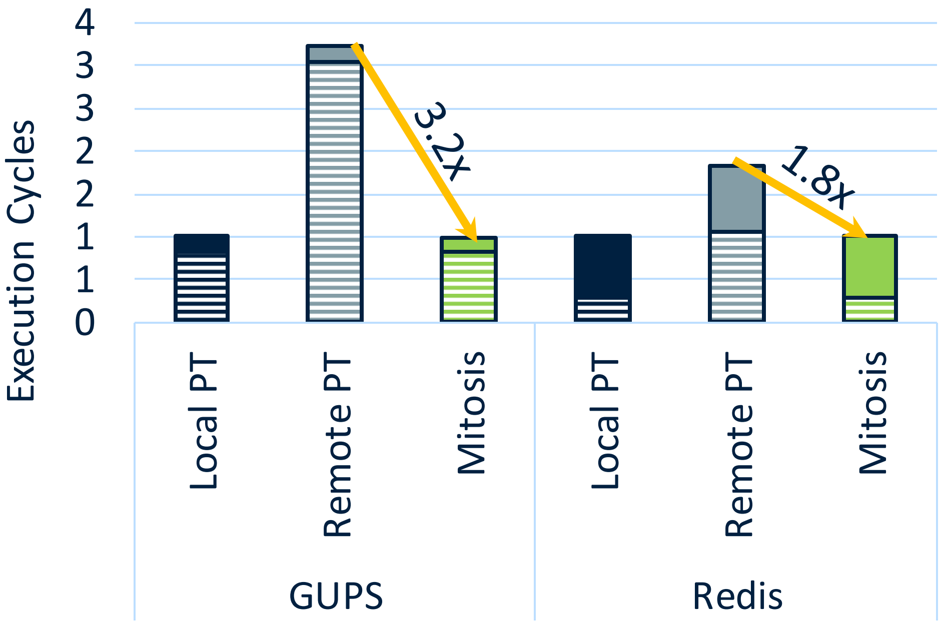


NUMA OBLIVIOUS



MITOSIS ON NATIVE LINUX



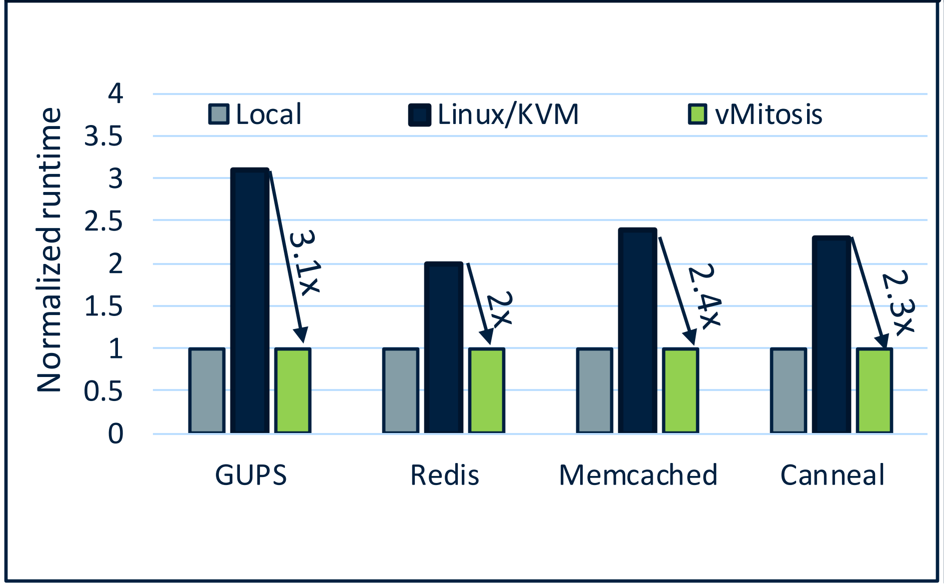


Mitosis **migrates** the page table

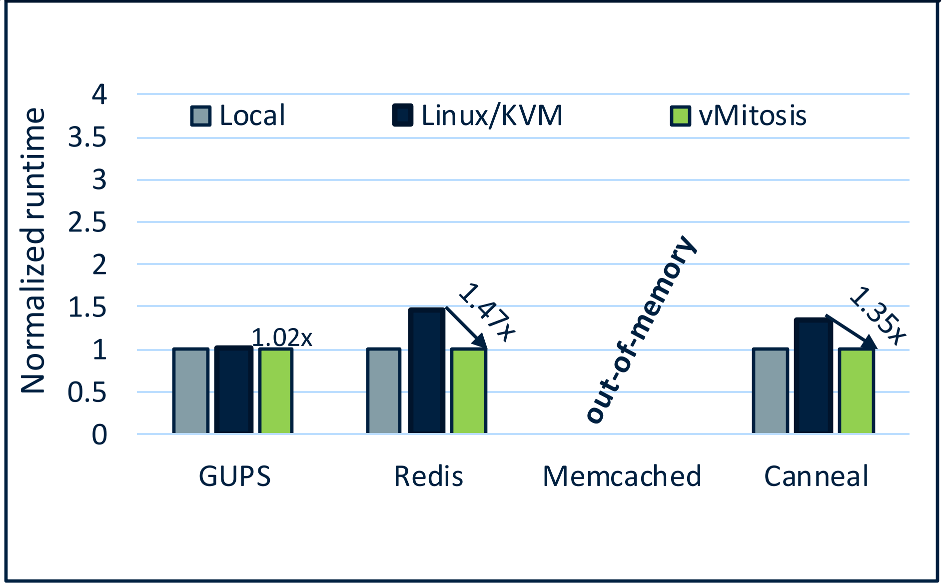
Workload-Migration Scenario

MITOSIS ON KVM VIRTUAL MACHINE

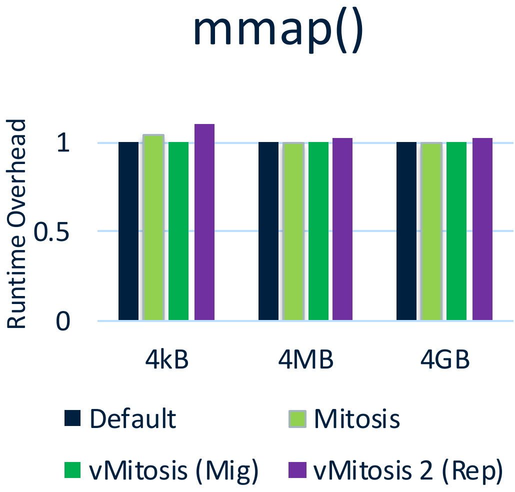
Page size: 4KiB

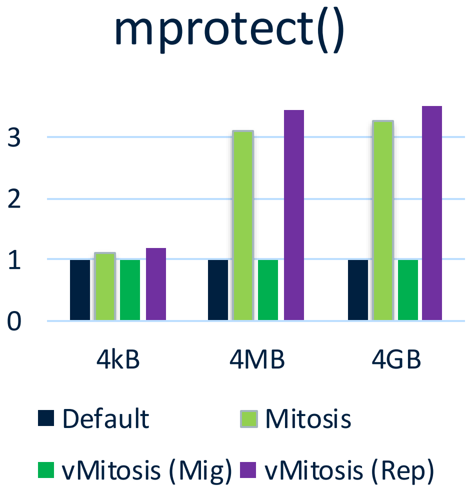


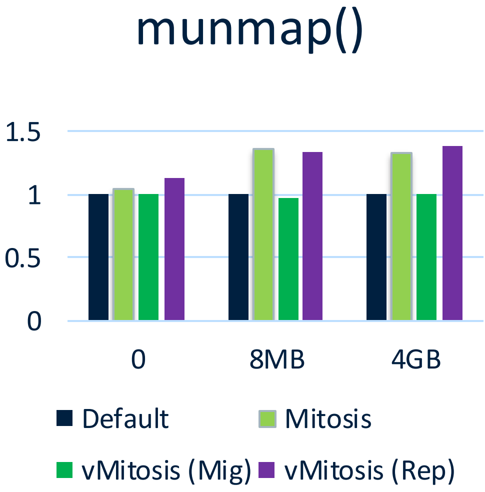
Page size: 2MiB (THP)



OVERHEAD OF MMAP AND FRIENDS







Effects of **eager** updates to 4 replicas – fixable with deeper page fault handler integration

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

ACKNOWLEDGEMENTS

Ashish Panwar

Arkaprava Basu

Kanchi Gopinath

Abhishek Bhattacharjee

Timothy Roscoe

Jayneel Gandhi

MITOSIS: ELIMINATE NUMA EFFECTS ON PAGE TABLE WALKS

Observations	Contributions	Results
Large memory server use **multi-socket architecture**.	**Systematic study** showing NUMA effects on page table walks.	Speedup for big-memory workloads **without application modifications**
Existing NUMA optimizations **ignore kernel objects**.	Mechanisms and policy for **replication and migration** of page tables	1.06 - 1.6x speed up for multi-socket workloads
NUMA effects on **page table walks** are inevitable.	**Linux + KVM** implementation	1.8 - 3.1x speed up for single-socket workloads

Contact:
achreto@cs.ubc.ca

Available on GitHub
https://github.com/mitosis-project