Rebooting Virtual Memory with Midgard

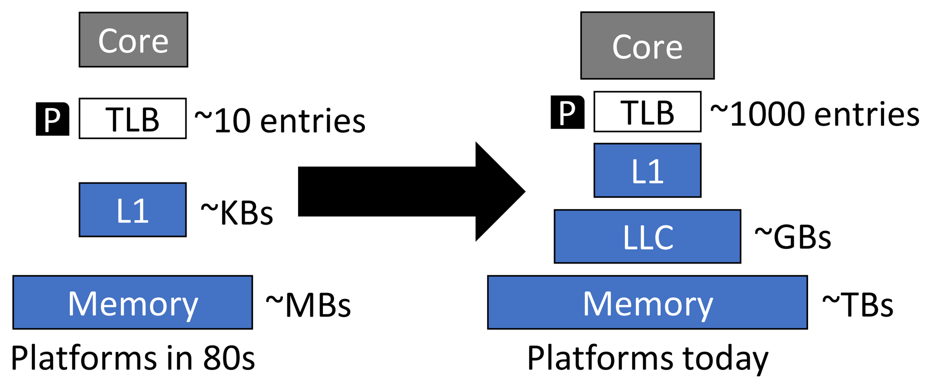
Siddharth Gupta, Atri Bhattacharyya, Yunho Oh‡, Abhishek Bhattacharjee†, Babak Falsafi, Mathias Payer

EcoCloud, EPFL ‡*Sungkyunkwan University* † *Yale University*

Virtual Memory Performance Pitfall

Increasing memory capacity puts pressure on TLBs

- Servers feature TB-scale memory hierarchies [AWS, GCloud]
- TLBs provide only MB-scale coverage with thousands of entries/core
- Frequent, long-latency page table walks are a performance bottleneck



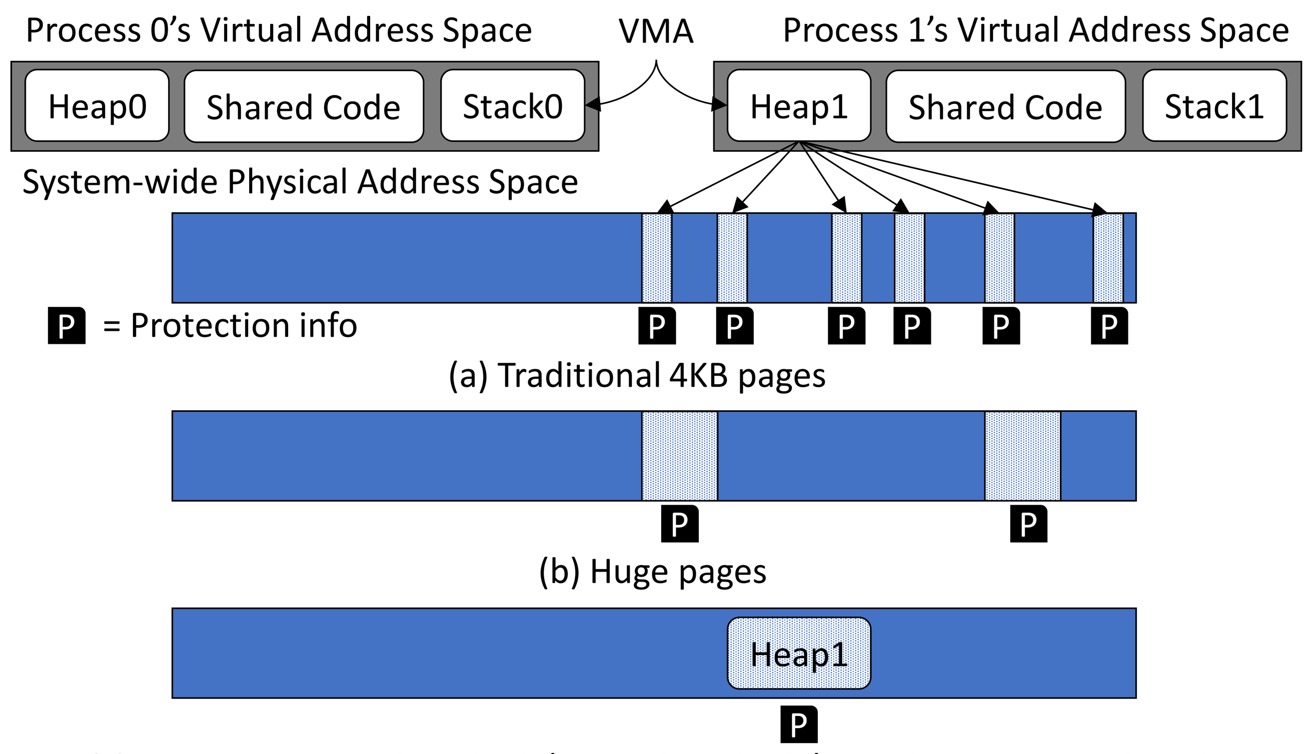
Product	Year	Cores	Cache capacity	TLB entries	Coverage (4KB)
Intel P4	2000	1	256KB SRAM	64	256KB
Intel KabyLake	2016	4	128MB eDRAM	1536	6MB
Apple M1	2020	8 (4+4)	16MB SRAM	3096	48MB (16KB)
AMD Zen3	2021	64 (8x8)	256MB SRAM	2048	8MB
Intel Sapphire Rapids	2022	56 (14x4)	64GB HBM2	?	?

TLBs cannot scale with memory capacity in the post-Moore era

Data Contiguity Can Help

But huge pages are not a panacea

- Huge pages increase TLB entry coverage and require physical contiguity
- But cause memory fragmentation by introducing multiple page sizes



(a) Traditional 4KB pages

(b) Huge pages

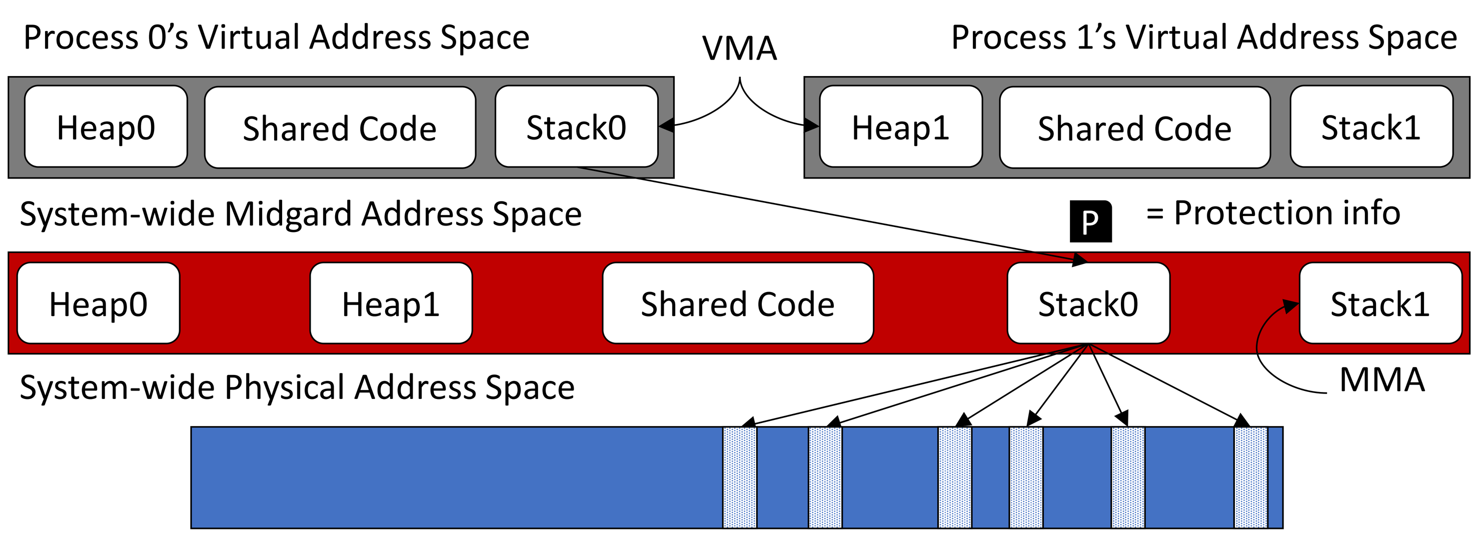
(c) Direct segments [ISCA'13] / RMM [ISCA'15] / Translation Ranger [ISCA'19]

POSIX VMAs are the inherent source of data contiguity

Virtual Memory Areas (VMAs)

VMAs represent data sections present in address spaces

- Apps organize their address space into VMAs (e.g. heap, stack, code)
- Each VMA is divided and mapped to physical pages



Access control should be performed using the default VMAs

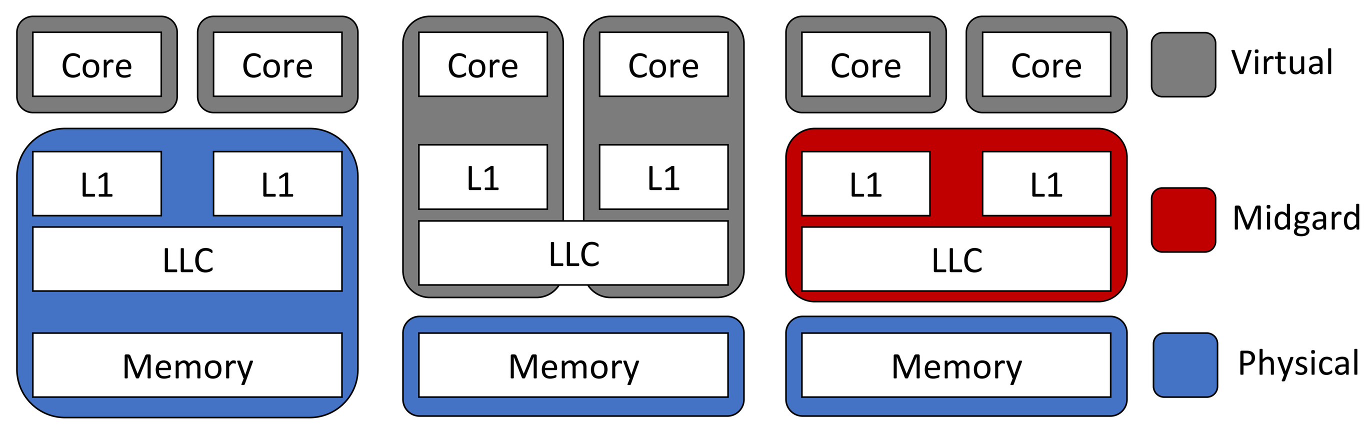
- Permissions are inherently defined at VMA granularity
- Need an intermediate address space to resolve synonyms!

Midgard address space contains a unique mapping of each VMA

Midgard: The Fundamental Divide

Midgard as an intermediate address space

- Logical and global address space managed by the OS
- Deduplicated VMAs are mapped to Midgard and then to pages



(a) Traditional. (b) Virtual Hierarchies. (c) Midgard.

Placement of address spaces in the memory hierarchy

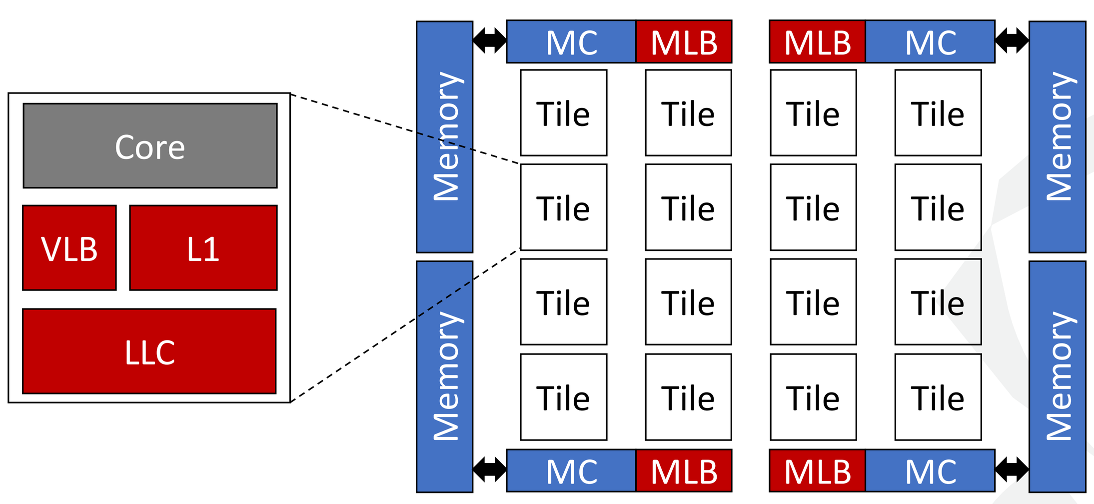
- Perform access control and cache hierarchy lookups at VMA granularity
- Perform memory management using page granularity

Midgard decouples application-based and OS-based characteristics

System Design

Frontside translation (VA→MA)

- Only 10s of VMAs per thread as working set
- Small and fast VLB per core with easy refills from a small VMA table



Backside translation (MA→PA)

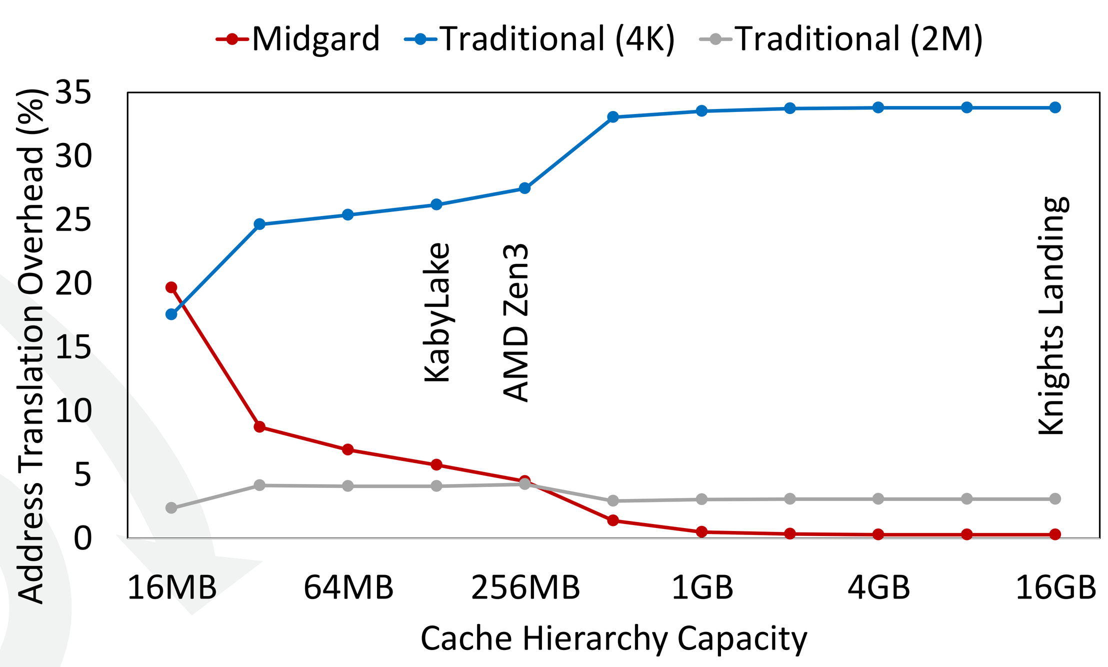
- Optimized page table walks using in-cache address translation
- Optional MLBs co-located with memory controllers for spatial locality
- Stock cache coherence protocol controls PTE copies (w/o MLBs)

Delaying slow translations provides flexibility and performance

Evaluation

Methodology:

- AMAT analysis with trace-based simulation using QFlex
- 16 cores with 1K baseline TLB entries running graph workloads



- Baseline overhead increases with cache hierarchy capacity
- Midgard enables the overhead to scale with the cache hierarchy capacity

Midgard future proofs VM by introducing VMAs in hardware

성균관대학교 1398 SUNG KYUN KWAN UNIVERSITY Yale EPFL EcoCloud