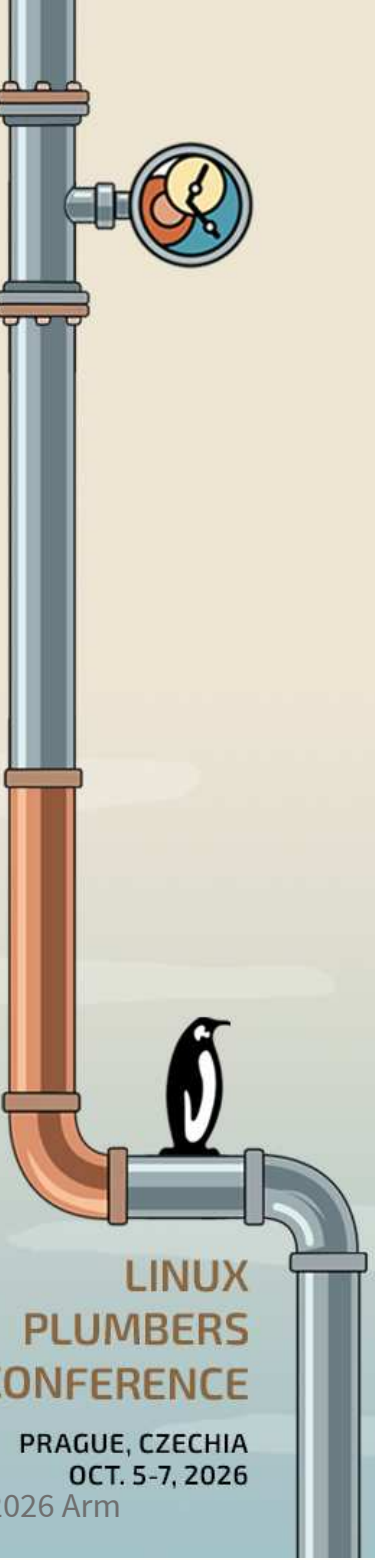


Direct map fragmentation (yes, again!)

Kernel Memory Management MC

Kevin Brodsky
arm



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

© 2026 Arm

Direct map — PTE only

 default mapping  modified attributes or validity



Direct map — PTE only

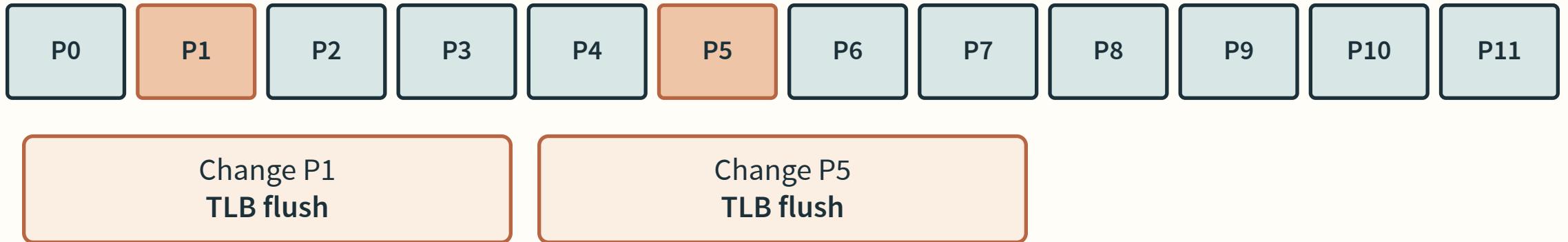
 default mapping  modified attributes or validity



Change P1
TLB flush

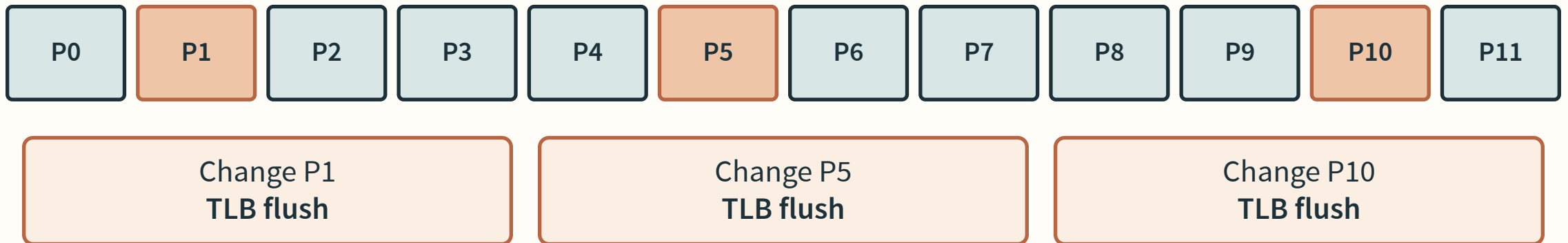
Direct map — PTE only

 default mapping  modified attributes or validity



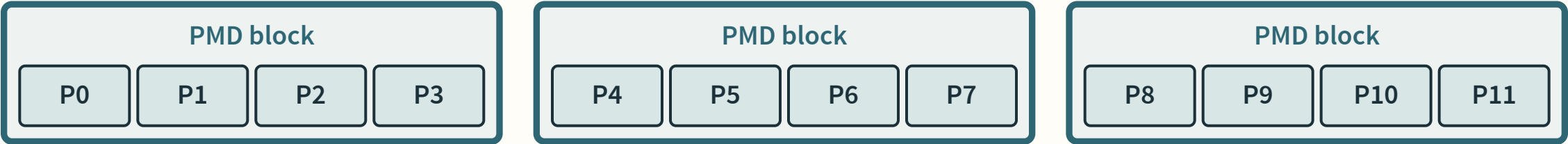
Direct map — PTE only

 default mapping  modified attributes or validity

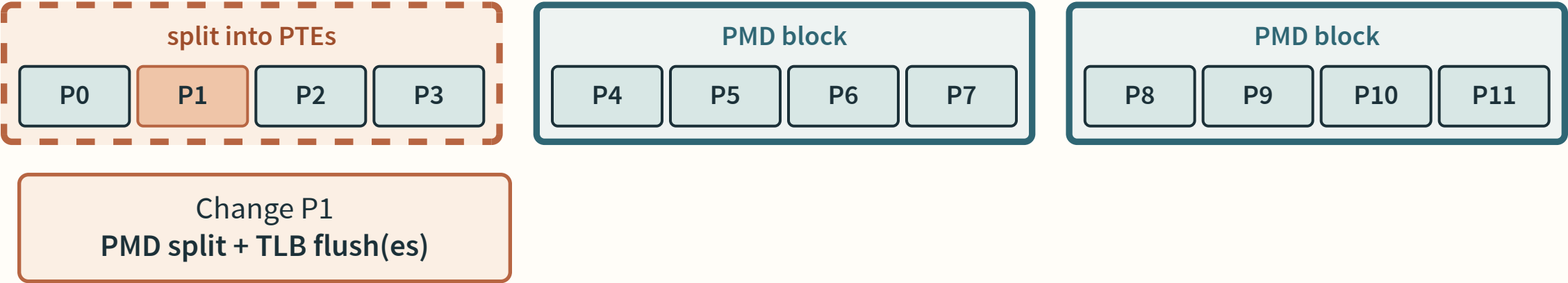


3 independent changes → 3 TLB flushes

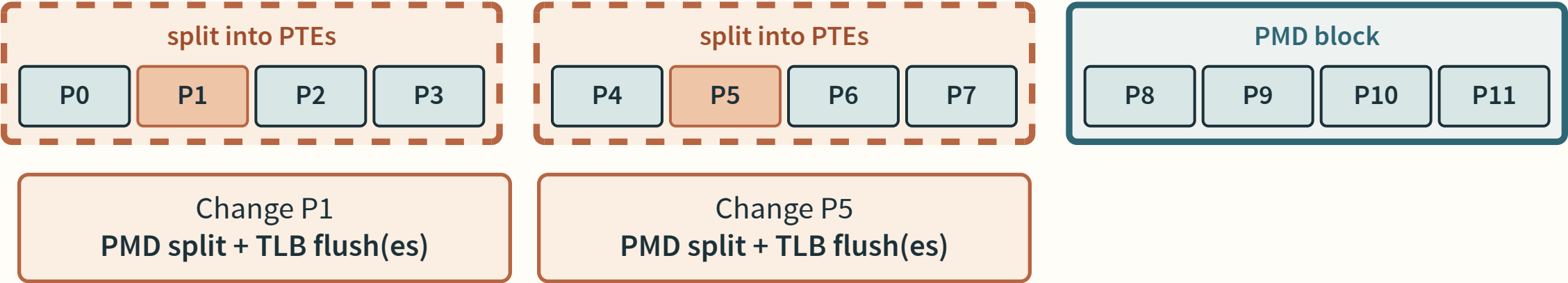
Direct map — PMD block mappings



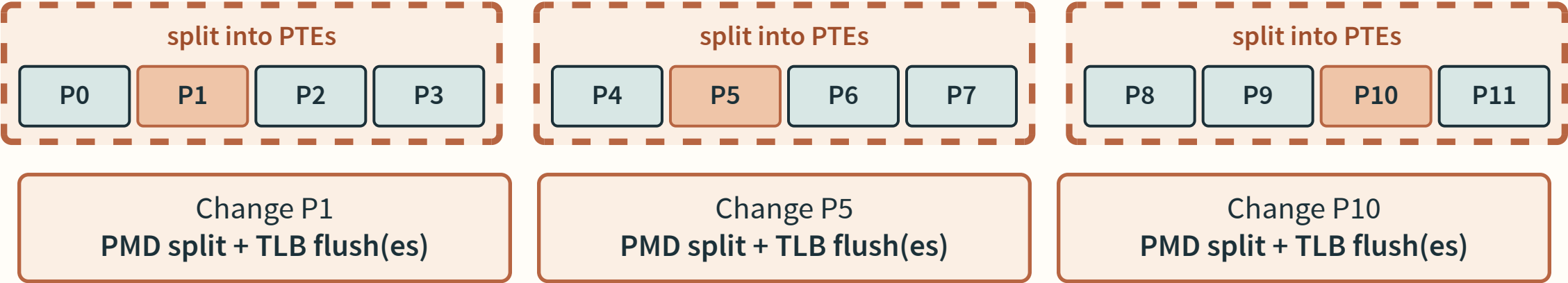
Direct map — PMD block mappings



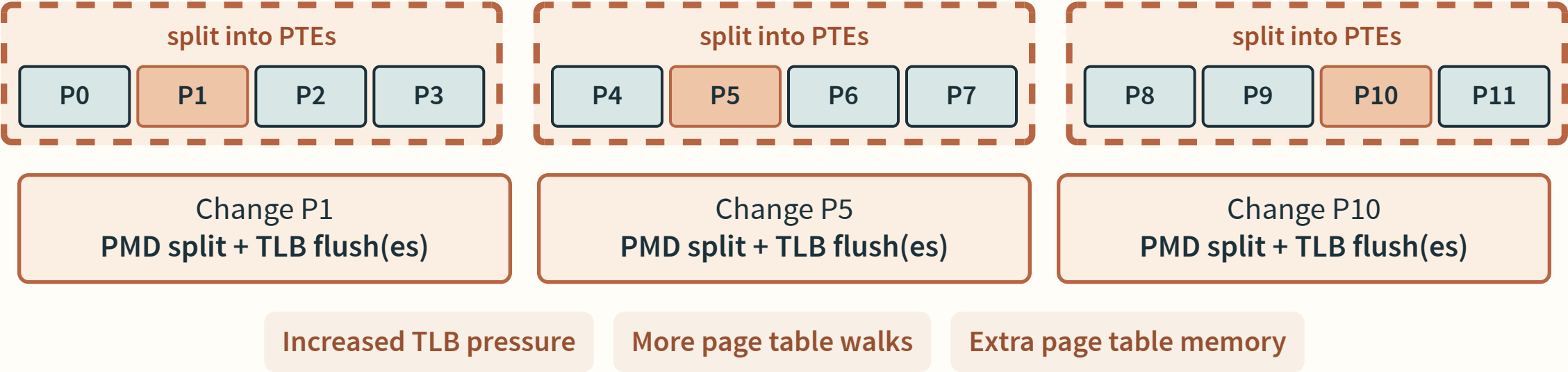
Direct map — PMD block mappings



Direct map — PMD block mappings



Direct map — PMD block mappings



Why remap individual pages?

Established users

- `execmem`: ensure direct map alias is **read-only**
- `memfd_secret`: faulted pages must be **removed** from the direct map
- Shared/decrypted buffers in confidential guests

Emerging motivations

- Direct map removal for `guest_memfd`
- pkeys-protected page tables
- Address space isolation

Depending on the use-case: **map/unmap**, or changing some **pgtable attribute**

All require dynamic, page-granular operations

Batching direct map changes

Today: only x86 `execmem` uses a cache.

Many proposals in recent years:

- 2021** PMD-sized `secretmem` pools
- 2021** Grouped `vma` alloc
- 2023** `__GFP_UNMAPPED` backed by dedicated cache
- 2026** Coalescing in: `secretmem`, arm64 `execmem`
- 2026** `ALLOC_UNMAPPED` expanding `migratetypes`

Examples from the [topic proposal](#)

Why batch?

- Amortise TLB flushes
- Potentially preserve block mappings

How much does fragmentation matter?

Mike Rapoport, LSF/MM 2023:

Little measurable impact on data accesses; even a fully fragmented direct map showed only single-digit degradation

Reconsidering the direct-map fragmentation problem [LWN.net]

Benchmarking: 4K vs large blocks

Baseline: PTE-only (4K) direct map
Percentage = improvement with PMD/PUD (2M/1G) block mappings

Benchmark	Measurement	m7i x86-64	m7g arm64	AmpereOne arm64
kernbench	System time	8.6%	10.6%	26%

Benchmarking: 4K vs large blocks

Baseline: **PTE-only (4K) direct map**
Percentage = improvement with **PMD/PUD (2M/1G) block mappings**

Benchmark	Measurement	m7i x86-64	m7g arm64	AmpereOne arm64
kernbench	System time	8.6%	10.6%	26%
	System time · THP off	5.6%	10.5%	4.8%
hackbench	Sockets · 80+ groups	~5%	8–10%	9–14%
SQLite speedtest	Run time	3.9%	3.4%	5.3%
vmalloc	fix_size_alloc_test · 1 page	2.6%	14.4%	5.0%
	random_size_align_alloc_test · 1 page	4.1%	12.5%	11.4%

x86 · Amazon EC2 m7i.24xlarge

48 cores / 96 vCPUs · 384 GiB

- PTE baseline: page_size_mask=0
- Fedora config; KFENCE off
 - init-on-alloc + extra hardening

arm64 · common setup

m7g.metal · 64 cores · 256 GiB (Graviton3)

AmpereOne · 192 cores · ~512 GiB

- rodata=noalias (to allow block mappings on m7g)
- PTE baseline: force_pte_mapping() always true
- defconfig

thp_anon=2M: always, except the labelled THP-off row.

4K vs blocks: not just about performance

Memory savings from skipping PTEs

8-byte PTE per 4 KiB mapped → ~2 MiB per GiB of RAM (0.2%)

m7i: ~768 MiB · m7g: ~512 MiB · AmpereOne: ~1 GiB

Power consumption

*“We've measured a considerable (double digit %) increase in **CPU power consumption** for some real-life camera workloads when mapping the linear map at 4k granularity thanks to the **additional memory traffic from the PTW.**”*

Will Deacon, replying to RFC v5 pkeys-based page table hardening

Discussion

Do we need a generic solution to avoid fragmentation?

Does it matter enough?

Are the performance, memory and power overheads sufficient? What else should we measure?

Is it actually possible?

Tricky aspects such as recursive page table allocation

Are subsystem caches good enough?

Risk of duplication with subtle differences (e.g. coalescing)

Promising efforts

mm: Add `ALLOC_UNMAPPED` and `AS_NO_DIRECT_MAP`

→ **First-class awareness** in the buddy allocator; direct map state reflected in `migratetype/freetype`
[Only supports mapped/unmapped states currently]

arch, mm: create a generic `set_memory/change_page_attr` core

→ Generic `set_{memory,direct_map}` logic; brings currently x86-only features like **coalescing**

arm

Merci

Danke

Gracias

Grazie

谢谢

ありがとう

Asante

Thank You

감사합니다

धन्यवाद

Kiitos

شكراً

ধন্যবাদ

ధన్యవాదములు

Köszönöm

LINUX
PLUMBERS CONFERENCE

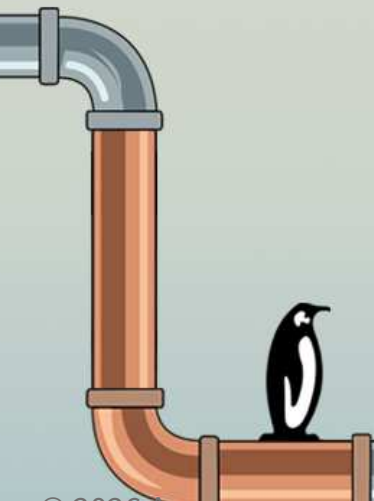
PRAGUE, CZECHIA
OCT. 5-7, 2026



arm

The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

www.arm.com/company/policies/trademarks



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

