



Why RISC-V Needs Larger Base Page

From PTE folding and mTHP to PGCL and native larger pages

Yunhui Cui · ByteDance · Linux Plumbers Conference 2026

Why 4 KiB Becomes a System-Scale Cost

At 1 TiB of RAM

BASE-PAGE OBJECTS

16×

≈15 GiB less metadata per TiB (64 B struct page)

LAST-LEVEL PTE MEMORY

16×

2 GiB at 4 KiB → 128 MiB at 64 KiB

Metadata

fewer descriptors

Fault work

fewer repeated operations

LRU / reclaim

fewer managed objects

Translation

larger TLB reach

A larger translation helps the MMU; a larger managed page helps the whole kernel.

Which Layer Is Actually Getting Larger?

USERSPACE PAGE SIZE

getpagesize · mmap offsets · mprotect · ELF · JIT layout

64K?

MANAGED-PAGE SIZE

allocator · folio · metadata · LRU · page cache · I/O

64K?

TRANSLATION SIZE

leaf PTE · NAPOT group · TLB reach · page walk

64K?

PTE folding gives us a larger TLB mapping; a larger managed page changes kernel memory management.

PTE Folding: 16 PTEs → 64 KiB

ordinary logical PTEs



64 KiB Svnepot translation
range

PUBLIC HELPERS

Logical 4 KiB view

`ptep_get()` · `set_ptes()`

RAW ARCH VIEW

Raw NAPOT view
for architecture code

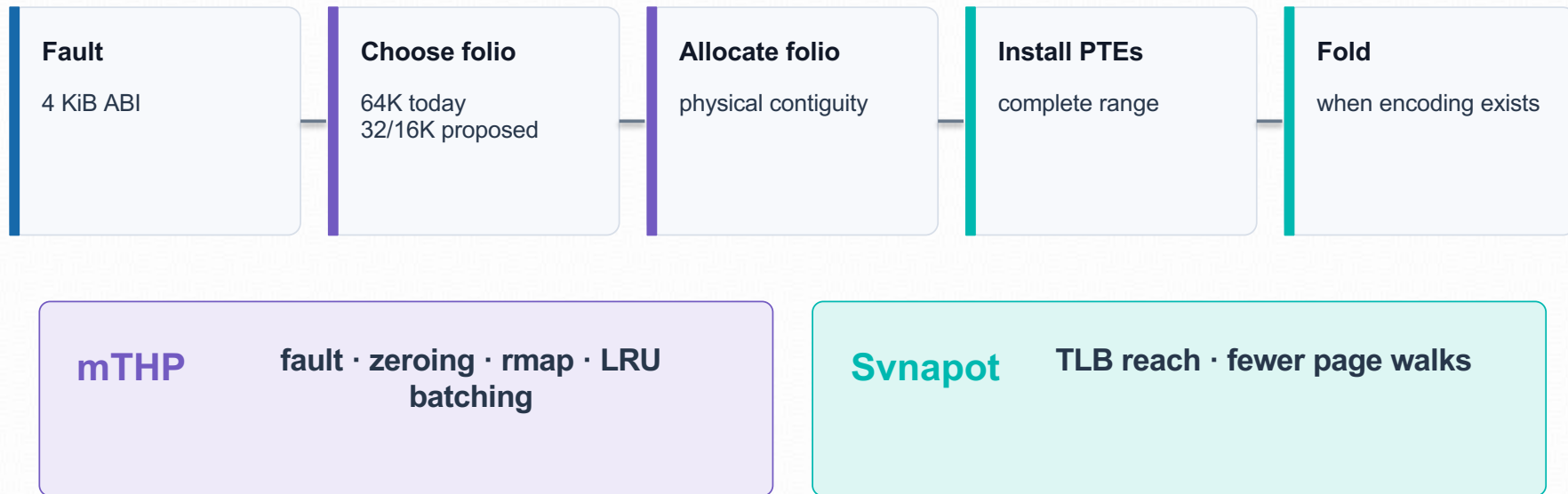
PARTIAL CHANGE

unfold before
`mprotect` · COW · `unmap`

[PTE folding v2 patchset](https://lore.kernel.org/all/cover.1784201104.git.cuiyunhui@bytedance.com/) → lore.kernel.org/all/cover.1784201104.git.cuiyunhui@bytedance.com/

The representation changes; Linux's per-4-KiB MM semantics must not.

mTHP Creates Foldable PTE Groups



mTHP is the producer; PTE folding is the translation consumer.

PTE Folding Cuts Translation Cost

Svnapot OFF → ON

lat_mem_rd

OFF

35.633 ns

ON

30.145 ns

15.40% faster

DTLB miss / access

OFF

7.202%

ON

0.910%

87.37% lower

SPEC 505.mcf_r

3.21% shorter

run time: 282.68 s → 273.61 s

mcf DTLB RATIO

-83.15%

miss/access reduction

REDIS CONTROL

-0.29%

throughput; within run-to-run noise

Lower miss rates help only when translation is on the critical path.

Why 64 KiB Folding Is Still Best Effort

POLICY

size / VMA policy rejects it

VMA WINDOW

64K window crosses an edge

PTE RANGE

entry / hole / attributes differ

RESOURCES

order-4 or memcg fails

Mapping-level reasons

ELF / file-offset phase

JIT RW → RX

mprotect / unmap

guard page / VMA edge

reclaim / deferred split

Any gate fails

64K

32K

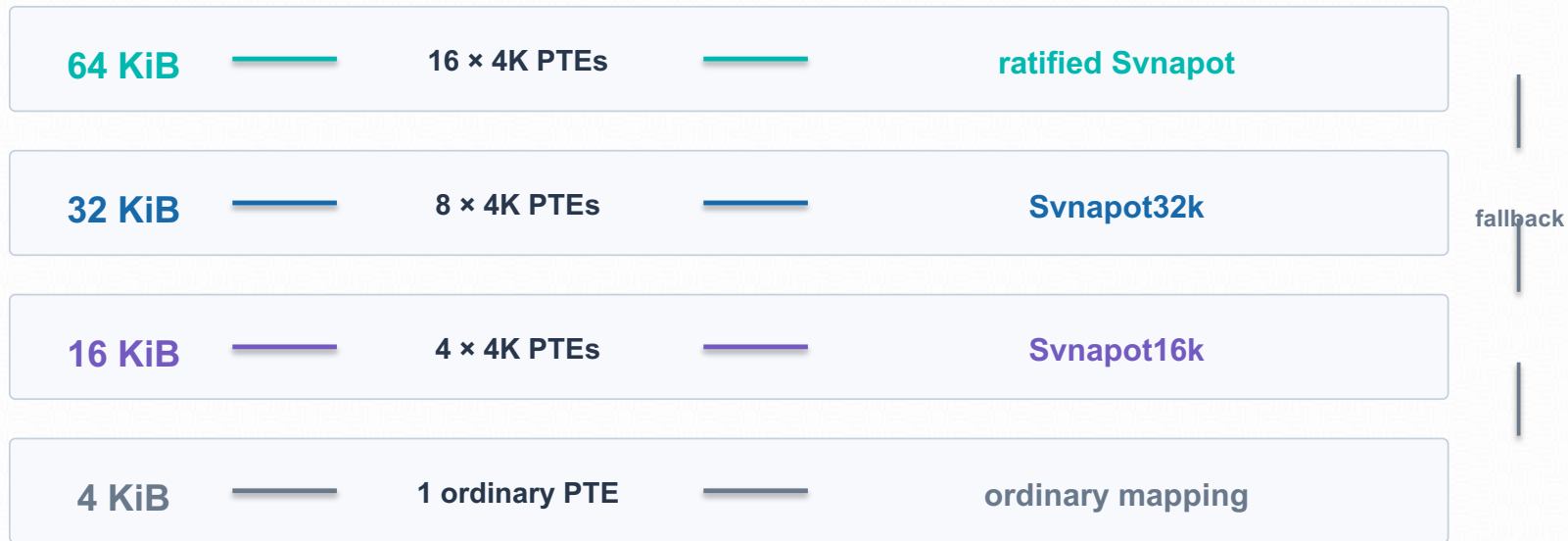
16K

4K

fallback ladder

Folding is opportunistic: policy, resources and mapping semantics must all agree.

Svnapot16k/32k: Proposed Fallback Steps

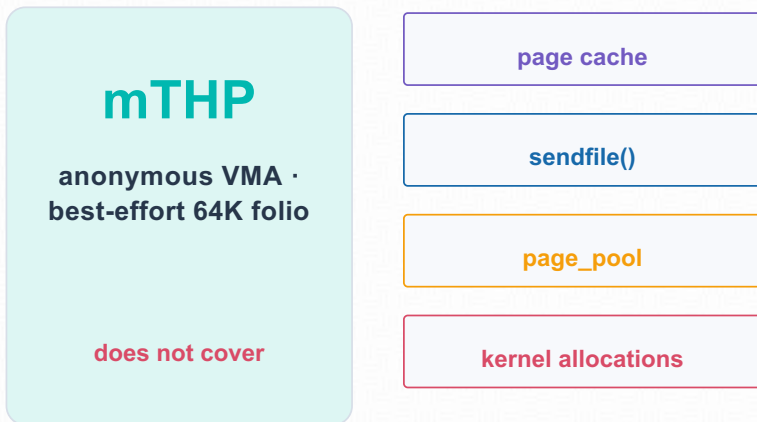


[Proposal and discussion → lists.riscv.org/g/tech-arch-review/topic/120745921](https://lists.riscv.org/g/tech-arch-review/topic/120745921)

If a 64 KiB fold is not possible, 32 KiB and 16 KiB encodings still provide larger TLB mappings before the final 4 KiB fallback.

Why Kernel Paths Need a Larger Order-0 Page

1 · mTHP does not cover every kernel path



2 · Why order-4 is not an equivalent production substitute



A larger order-0 page removes the high-order allocation dependency from these kernel paths.

Larger Kernel-Managed Pages with PGCL

PGCL, proposed by Kirill A. Shutemov · 64K kernel management · 4K userspace ABI

KERNEL MANAGEMENT

PG_SIZE = 64 KiB
buddy · folio · LRU
page cache · sendfile

USERSPACE CONTRACT

PTE / ABI = 4 KiB
getpagesize() = 4096
4K mprotect / munmap

TLB MAPPING

Svnepot → 64 KiB
only for a complete, aligned
and compatible PTE group

NGINX THROUGHPUT VS 4K

64 KiB response

mTHP +0.70%

PGCL +18.33%

1 MiB response

mTHP +0.67%

PGCL +19.26%

WHY THE 1 MiB CASE IS FASTER

sendfile calls

1 → 1

pipe entries

256 → 16

internal TCP calls

~17 → ~2

worker CPU/request

-45%

Same 1 MiB of data, handled as fewer and larger kernel chunks.

PGCL reduces per-page kernel work; with a 4K ABI, 64K Svnepot mappings remain best effort.

Arm64: mTHP + Contiguous PTE vs Native 64 KiB

Nginx · 1 MiB response · RPS

4K + mTHP



native 64K



native 64K: +24.85% vs mTHP

8 wrk threads · 256 connections · 60 s

MySQL

2.03–3.41%

mTHP below native
across RO / RW / WO

MULTI SPAWN

28.43%

mTHP below native 64K

MULTI EXECL

39.82%

mTHP below native 64K

RESULT SCOPE

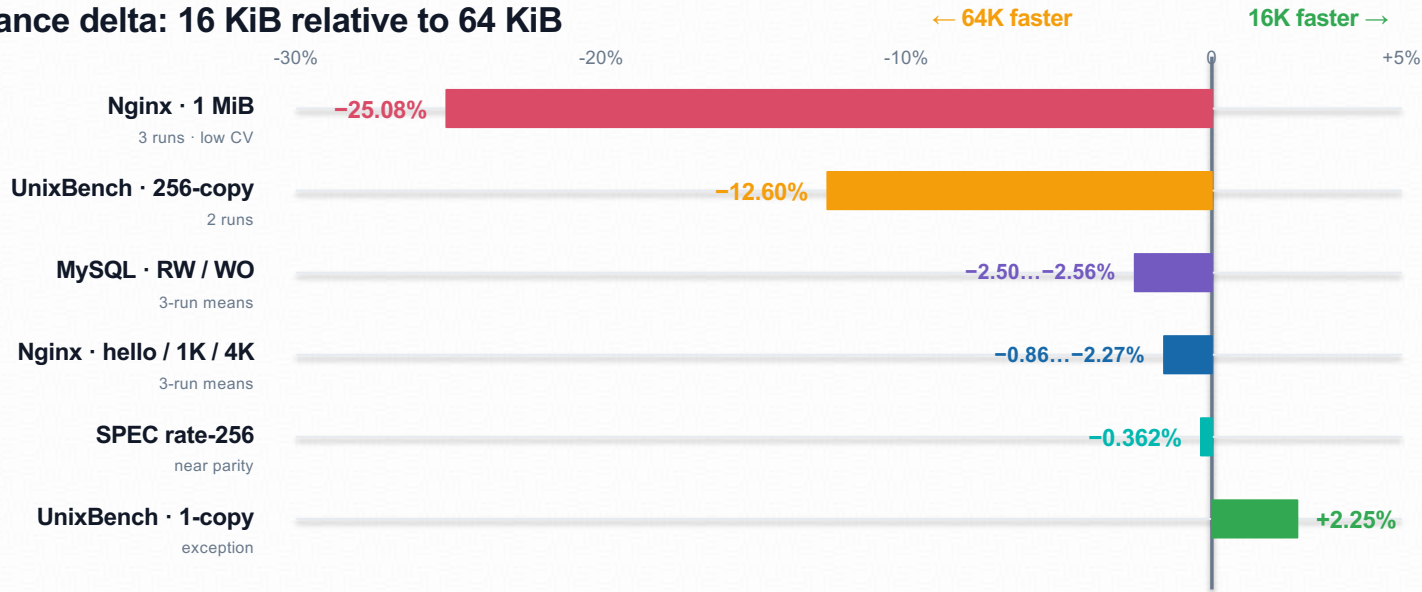
SYSTEM

not TLB-only

Equal nominal TLB reach does not imply equal system behavior.

Arm64: 16 KiB vs 64 KiB Across Workloads

Performance delta: 16 KiB relative to 64 KiB



MySQL RO -5.51% is excluded from the chart pending a symmetric rerun (64K CV 6.10%).

Across measured workloads, 16 KiB shows no overall advantage; 64 KiB leads in large-response and high-concurrency cases.

Arm64: Native 64 KiB Memory Cost and Fragmentation

1 · MEASURED CGROUP MEMORY FOOTPRINT

MySQL · THP-off **+1.13–1.36%** 64K vs 4K

Redis **+0.59–2.24%** 64K vs 4K

Nginx **+6.15–9.00%** $\approx$ 2–10 MiB absolute

anon + file + kernel · steady-state median · not process RSS

2 · EXTERNAL-FRAGMENTATION SIGNALS

mm_page_alloc_extfrag

4K 188

16K 16

64K 1

event count—not a fragmentation rate

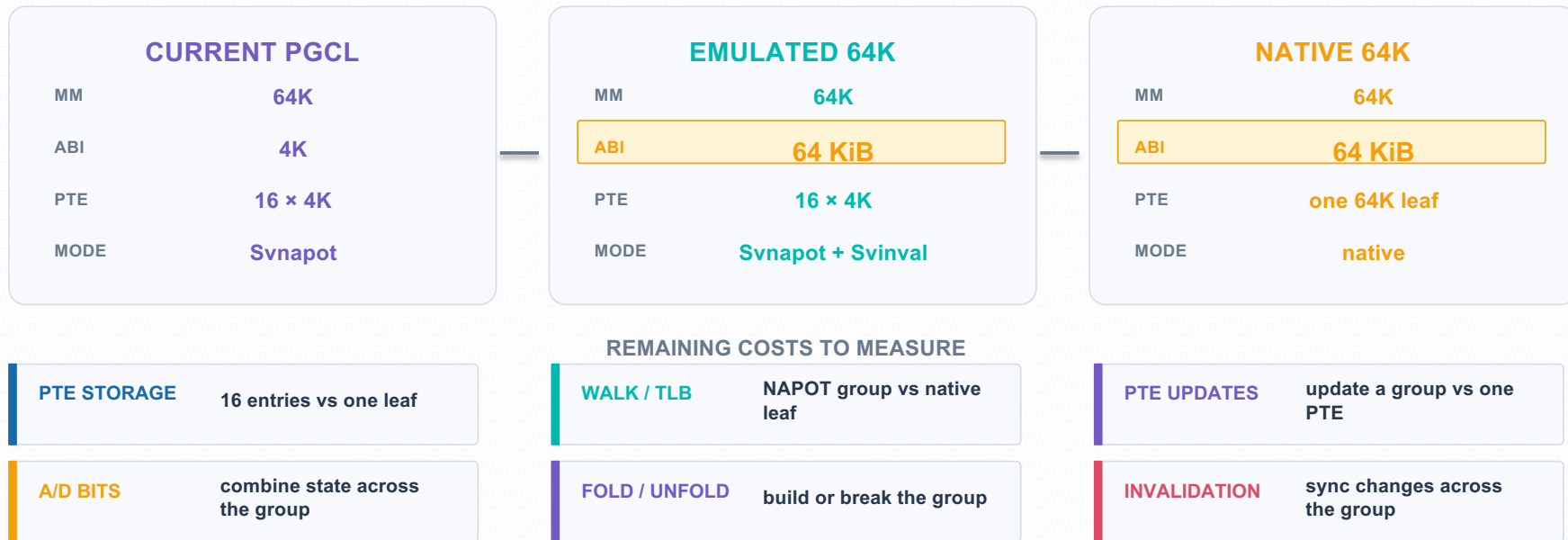
compaction = 0 · direct reclaim = 0 · allocstall = 0

Still missing: long-duration aging · sustained memory pressure · fallback/compaction · allocation latency

No 64 KiB fragmentation problem is visible here; long-duration stress is still required.

RISC-V: Gaps Between Emulated and Native 64 KiB

First match both models at 64 KiB MM + ABI; then compare sixteen NAPOT PTEs with one native leaf.



With the same 64 KiB ABI, native 64 KiB removes the 16-PTE group and its extra work.

Larger Base-Page VM TG: PoW and Decision Path

STATUS · TG formation proposal in progress · PoW v0.2

1 · DEPLOY NOW

mTHP +
PTE folding

Equivalent
models

Sail · QEMU · gem5 · Linux

2 · COMPLETE FALLBACK

Svnapot16k /
Svnapot32k

Decision
evidence

3 · TG PoW: MODEL & COMPARE

native 16K / 64K
vs emulated 16K / 64K

performance · fragmentation · ABI · toolchain · KVM ·
IOMMU

4 · DECIDE

new native mode or
existing-ISA guidance

Recommendation: advance native 64K; use the TG to validate the remaining details.

Use today's mechanisms while advancing native 64K through the TG.

Questions & Discussion

How should RISC-V provide larger base pages?

PTE FOLDING

lore.kernel.org/all/cover.1784201104.git..

⌵

click to open

Svnapot16k / 32k

lists.riscv.org/.../120745921

click to open

LARGER BASE-PAGE RFC / PoW

lists.riscv.org/.../120899216

click to open

Thank you