

Heterogeneous RISC-V Cores in Linux



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

Mark Zhuang & Dong Xu



SPACEMIT
进迭时空

LPC 2026 · RISC-V Microconference · 2026-10, Prague

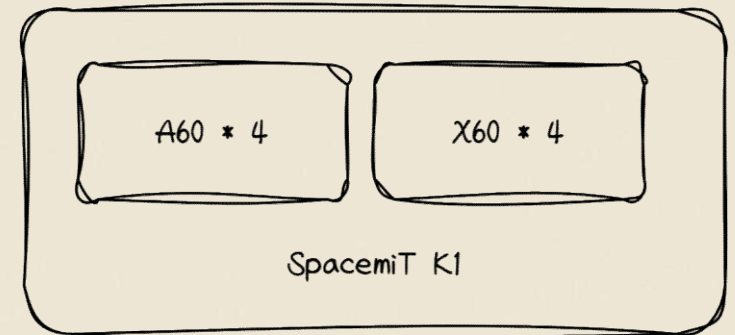
SpacemiT K1 — the easy case

8 cores, two quad clusters:

Cluster 0 (AI): 4 cores + xsmtvdot

Cluster 1 (GP): 4 cores — no xsmtvdot

Same VLEN, same base ISA



$A60 = X60 + \text{xsmtvdot extension}$

Upstream kernel:

Treat K1 as 8 identical cores

Vendor kernel:

On illegal-instruction trap for xsmtvdot → migrate task to Cluster 0

SpacemiT K3 — the hard case

H extension, vendor extension is presence/absence

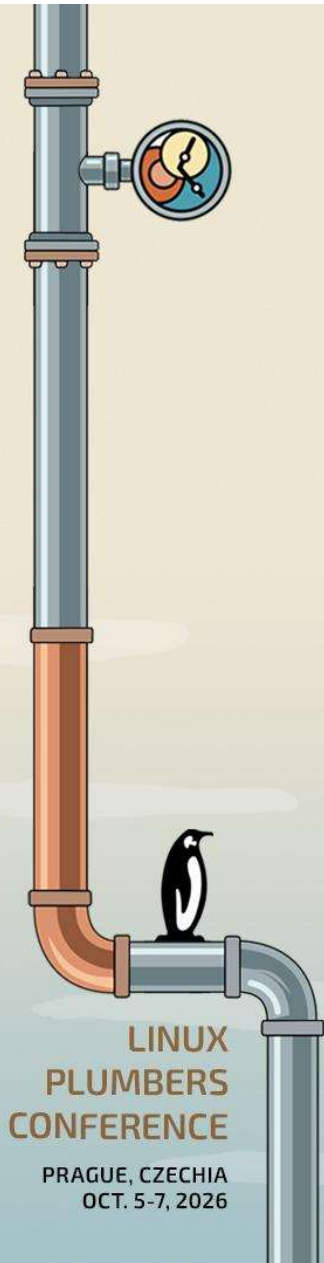
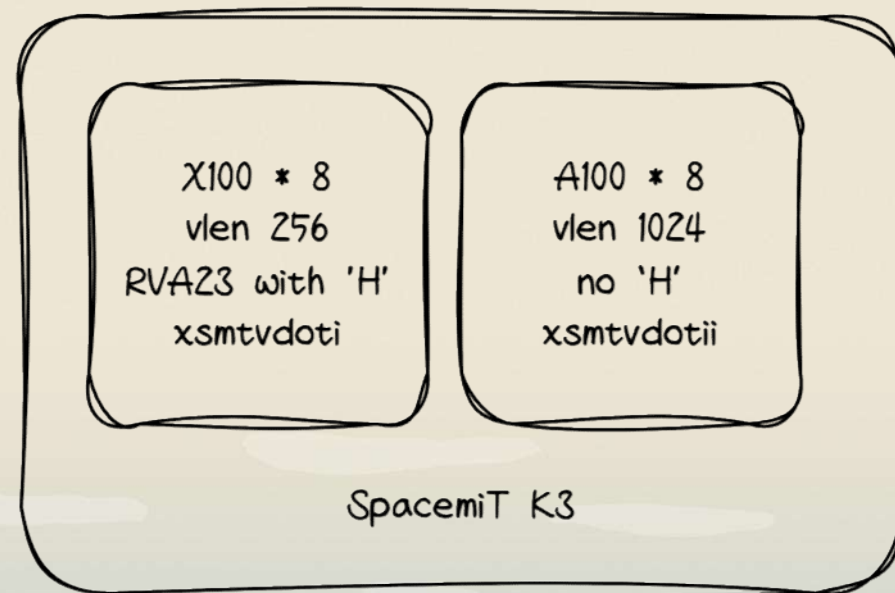
VLEN mismatch 256 vs 1024, semantic incompatibility

Upstream kernel:

Use 8 X100 cores only

Vendor kernel:

CONFIG_SPACEMIT_HMP to enable all 16 cores



Mismatched VLEN is hard to support

RVV is variable-length, but assumes a fixed VLEN at runtime

vsetvli output depends on hardware VLEN → same binary, different behaviour on X100 vs A100

After the first vector instruction:

All rvv register contents are semantically bound to VLEN

Migration to a different-VLEN core: save/restore undefined, data layout breaks

ARM big.LITTLE worst case: slow

RISC-V VLEN worst case: wrong



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

Vendor HMP — CONFIG_SPACEMIT_HMP

riscv_v_vsize = 4096 (hard-coded max across both core classes)

thread_type added to struct task_struct (HMP_REGULAR / HMP_AI)

cpu-ai = "true" in device tree — tags A100 CPU nodes

SoC driver maintains regular_cpu_mask / ai_cpu_mask

/proc/set_ai_thread — bind PID to AI cores at runtime

drivers/misc/tcm.c depends on SPACEMIT_HMP in Kconfig

<https://github.com/spacemit-com/linux-6.18>

Vendor SDK only — not suitable for upstream



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

HMP in practice: llama.cpp on K3

Bind llama.cpp to A100 (VLEN 1024) via /proc/set_ai_thread

For Qwen3-4B:

X100: 19 tokens/s@PP512, 5.1tokens/s@TG512

A100: 83 tokens/s@PP512, 11.2tokens/s@TG512

SpacemiT AI runtime for RISC-V

Spine-Runtime targets SpacemiT RISC-V SoCs and coordinates parallel tasks between general-purpose host cores and AI compute cores.

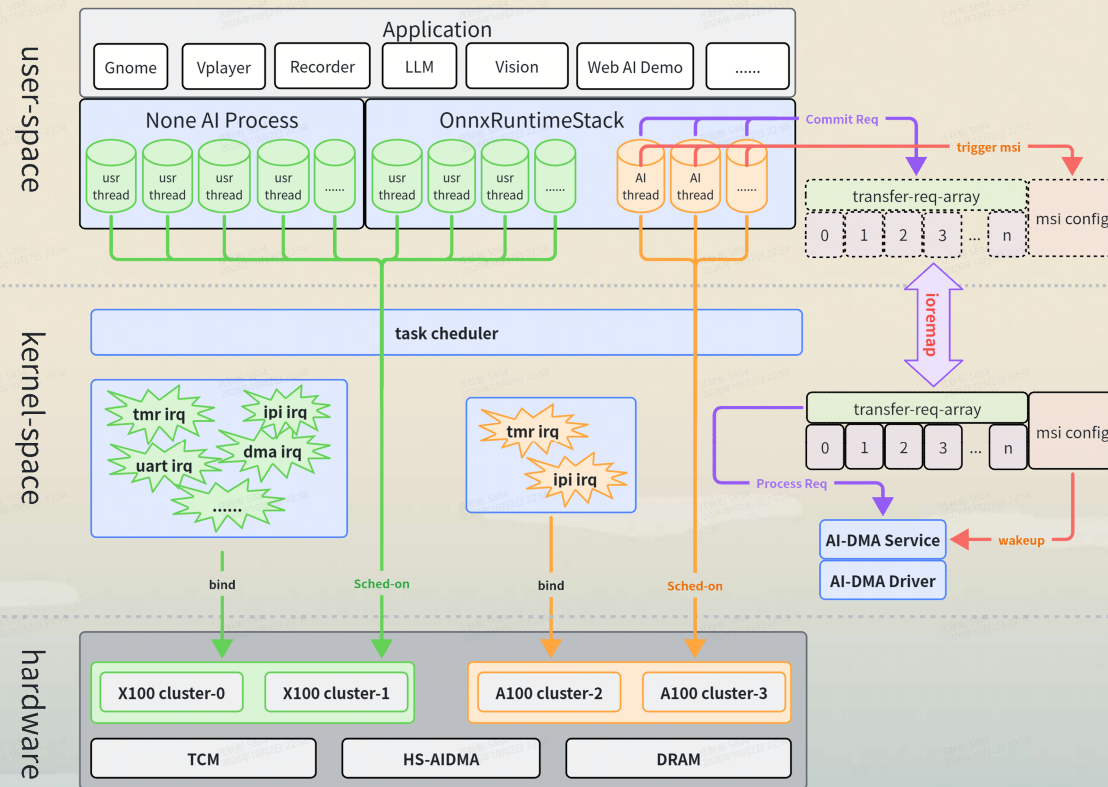
<https://github.com/spacemit-com/spine-runtime>



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

Vendor HMP



All threads and IRQs default to X100. Only core-dedicated and user-bound threads run on A100.

User must guarantee migration safety — the precondition for stable HMP.

https://github.com/brucehoul/k3_ai

https://github.com/zqb-all/k3_taskset



HMP limitation — and where this leads

Even with HMP: processes must be manually pinned to A100
No free kernel scheduling between X100 / A100

To relax that constraint, two directions:

(a) Software: binary translation lets a migrated process finish in-flight vector work
→ Chimera (next slide)

(b) Hardware: instruction to declare "vector state is clean — migrate me freely"
→ VINIT (future silicon)

Neither solves the upstream kernel problem



LINUX
PLUMBERS CONFERENCE PRAGUE, CZECHIA / OCT. 5-7, 2026



Chimera (ISCAS — EuroSys'26)

User-space binary rewriting for cross-core migration

<https://dl.acm.org/doi/10.1145/3767295.3769323>

Does not solve the upstream kernel problem



LINUX
PLUMBERS CONFERENCE PRAGUE, CZECHIA / OCT. 5-7, 2026



Hardware paths forward

⚠ Not implemented on K3 — design extrapolation for future silicon

Configurable VLEN on A-series cores:

Hardware switch: VLEN 1024 ↔ 256

Upstream: configure A100 to 256 → standard 16-core SMP

Vendor: keep VLEN 1024 + HMP for full AI throughput

Gives the user a choice

VINIT instruction:

User-space marks vector state as Initial → opens a migration window

Kernel can freely schedule during that window

Only helps software you compile / control — not a general solution



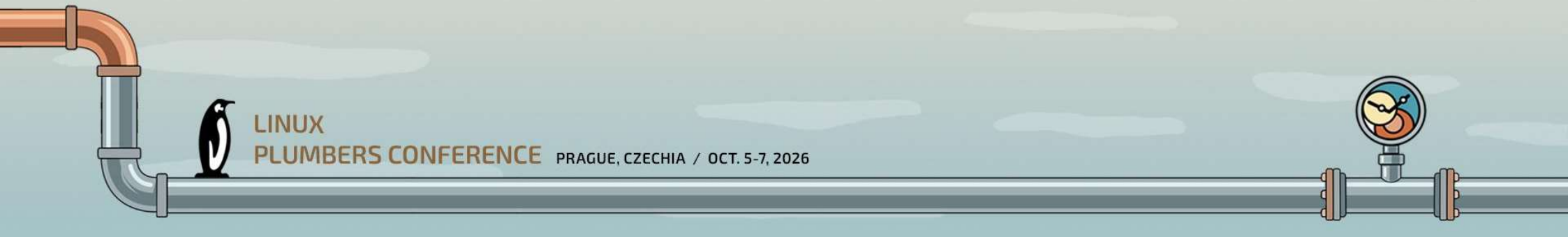
LINUX
PLUMBERS CONFERENCE PRAGUE, CZECHIA / OCT. 5-7, 2026



Other options

Run a separate Linux / RTOS on A100 entirely
Expose A100 as a compute accelerator (offload target for X100)

Both are viable; SpacemiT investment here is limited for now



Some open questions

Does upstream plan to support ISA-heterogeneous cores?

K1: xsmtvdot on some cores

K3: H extension on X100 only

K3: Mismatched VLEN

Should apps express CPU class preference to the scheduler?

"Prefer VLEN ≥ 1024 " / "I am vector-clean, migrate me freely"

Let the kernel decide, not force explicit process pinning



LINUX
PLUMBERS CONFERENCE PRAGUE, CZECHIA / OCT. 5-7, 2026



Thanks

SpacemiT looks forward to working with the community to explore potential upstreamable solutions for making full use of the compute capabilities of SoCs with heterogeneous RISC-V Cores.

