VATES

Xen Project

Xen on RISC-V

From dom0less to xl

Xen Summit 2026

Baptiste Le Duc
Hypervisor & Kernel Engineer, Vates

Oleksii Kurochko
Hypervisor & Kernel Engineer, Vates

Plan

1. **Presentation & context**
2. **Current upstream status**
3. **What we have in downstream**
4. **Future roadmap**
5. **Live demo**

Part 1

Presentation & context

Who are we?

- **Baptiste Le Duc**
  - Intern at Vates for 6 months working on the RISC-V port, now full-time **Hypervisor & Kernel Engineer**
  - Passionate about OSes and IoT
  - Eager to learn
- **Oleksii Kurochko**
  - **Hypervisor & Kernel Engineer** at Vates, mainly working on RISC-V related topics
  - Leading the Xen RISC-V port upstream
  - Enjoys low-level work, close to the hardware

Why RISC-V?

- Open, royalty-free instruction set (ISA)
- No licensing fees, unlike proprietary x86 and ARM
- Enables **full** open stack from **hardware** to **cloud datacenters**

→ future of European **digital sovereignty**

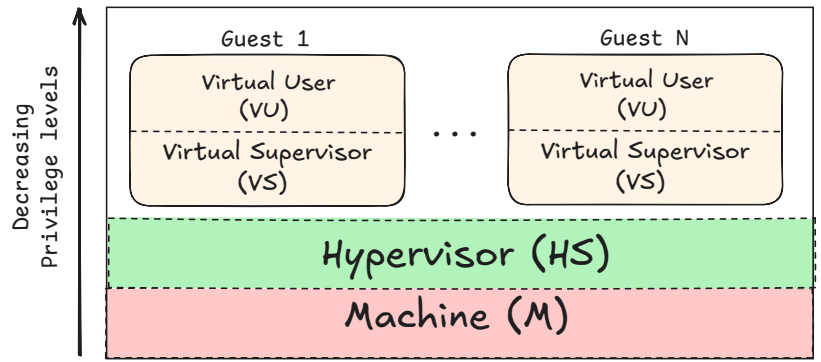


Figure 1: H-extension privilege levels

Target platform

We target a **server-class RISC-V** profile with hardware virtualization support:

Required

- **H-extension**: hardware virtualization, two-stage MMU
- **AIA / PLIC**: interrupt controllers
- **MMU**: two-stage address translation
- **Zbb extension**: bit-manipulation
- **Svpbmt extension***: memory attribute types
- **Zihintpause extension***: pause hint

Nice to have

- **IOMMU**: for device passthrough
- **Sstc extension**: for supervisor timer compare
- **Shtvala extension**: guarantee of having guest physical address of faulting instruction

→ all ratified in the RVA23 profile + server platform spec

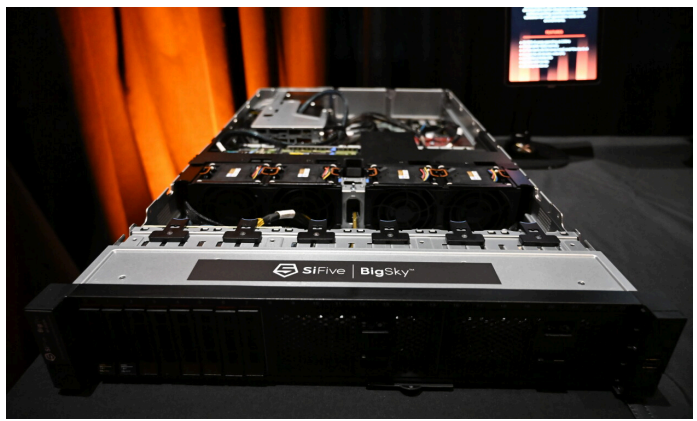


*Figure 2: **SiFive BigSky** (image credit: SiFive)*

** Required by Xen today, but only until the patches relaxing them land, Xen will fall back to a safe default instead of refusing to boot.*

Part 2

Current upstream status

Upstreaming order:

1. `dom0less` -> our first target

2. `dom0` + `xl` toolstack

Upstreaming now

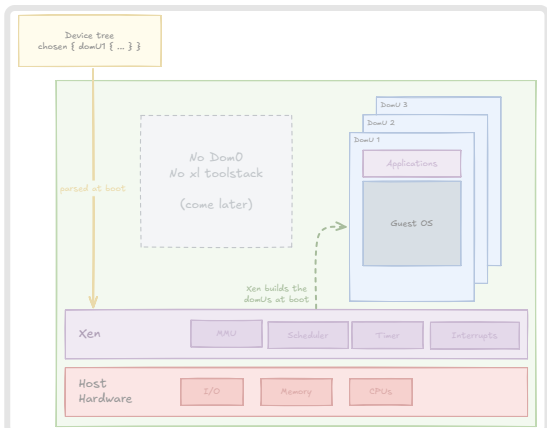


Figure 3: dom0less: Xen boots the domUs itself at boot (static config)

Comes after

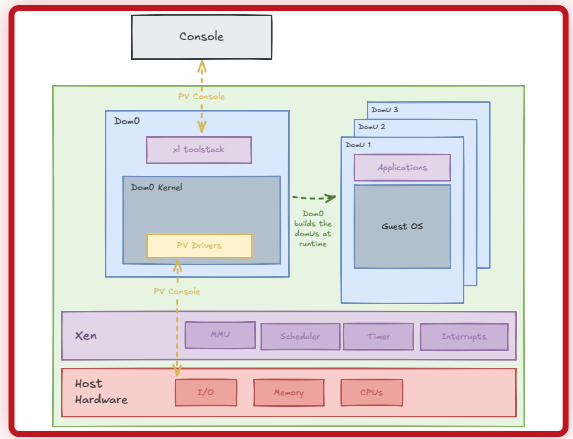


Figure 4: dom0 + xl toolstack: guests managed from dom0

What upstream Xen for RISC-V already has

Xen boots and runs

- **DTB loading / parsing**
- **CPU feature detection** refuses to boot if a required extension is missing
- **Console**: SBI early `printk`, then 16550 from DT
- **Trap handling** + exception tables
- **Interrupt support**: AIA only (APLIC + IMSIC) for Xen's own interrupts
- **Timers**

Guest infrastructure

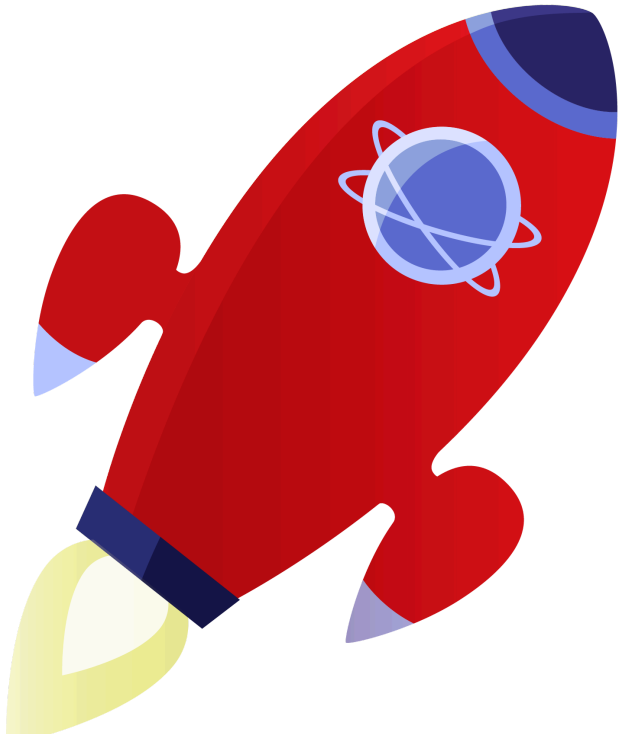
- **P2M handling**: guest physical → host physical, VMID, superpages, foreign mappings
- **vSBI framework**: base + legacy extensions only
- **vtimer**: per-vCPU guest timer
- **Linux `Image` loading** + guest RAM layout
- **`dom0less` guest domain creation** (not launch yet)*

… but Xen still **halts** right after setup: single CPU, no context switch, no hypercalls, no event channels, no grant tables

** Not merged yet at the time these slides were written. The patch series is already in its final form, so it will most likely be merged by the time of the conference.*

Part 3

What we have downstream



Downstream: what makes a guest actually run

Interrupts

- **vAIA**: virtual APLIC + virtual IMSIC
- **IMSIC interrupt files**: hardware VS-file + **software** fallback
- **vCPU migration**: interrupt files follow the vCPU
- **PLIC + vPLIC**: for pre-AIA hardware

Guest interfaces

- **vSBI extensions**: HSM, IPI, RFENCE
- **Device passthrough**: MMIO + IRQ routing, **no IOMMU** — prototype sits on an unmerged branch

Running guests

- **SMP**: secondary harts for Xen, several vCPUs per guest
- **vCPU context switch** + scheduling: idle loop, AIA and timer state

Emulation

- **MMIO trapping** + device-agnostic dispatch
- **Instruction decoding**: loads, stores, CSRs, WFI
- **Guest page faults** + trap redirection to the guest

guests **boot and run**, on QEMU and on real hardware

RISC-V is running on real hardware!

Goal: mostly marketing than a long-term commitment

→ Got published in the **RISC-V Weekly Newsletter** by RISC-V International

HiFive Premier P550

- Recent board (late 2024), but an old core (June 2021)
- Old **H-extension** (v0.6.0)
- **PLIC** interrupt controller

→ Needed some adaptations

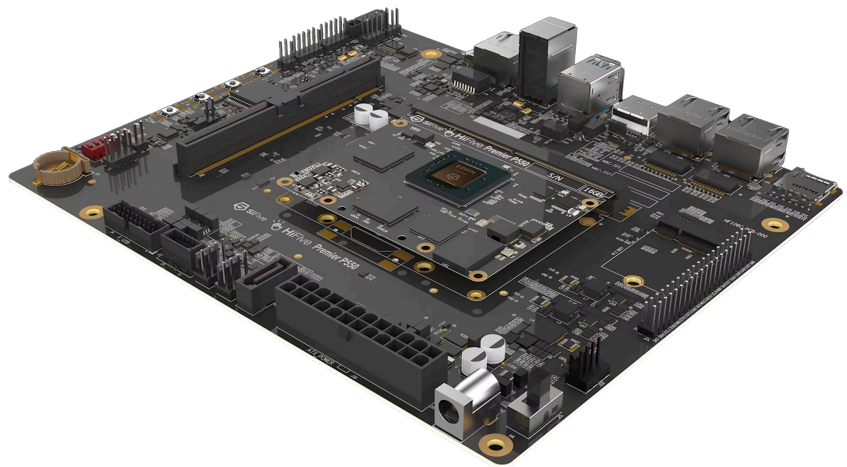


*Figure 5: **HiFive Premier P550***
(image credit: SiFive)

New CI smoke test added

RISC-V is running on real hardware!

SpaceMIT K3

- Only board found supporting **both H-extension and AIA** interrupt controller

➡ Got booted by a **community contributor**

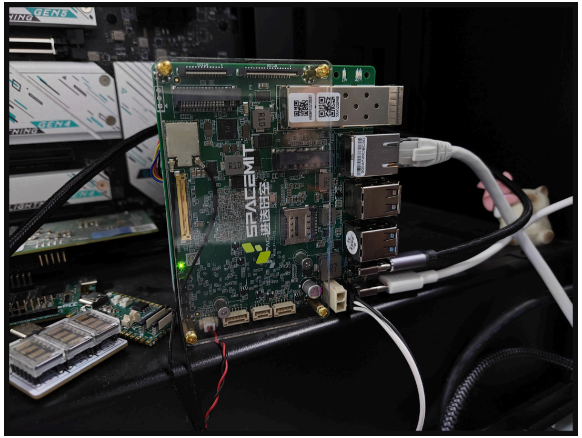


*Figure 6: **SpaceMIT K3***
(image credit: Zheng Zhang)

*Figure 7: **Board's logs***

Dom0

- **Hypercalls**
- **Device Tree generation**: describes the hardware to the Linux kernel. `xl` generates the domU's DT
- **Linux guest enlightenment**: the kernel detects Xen at boot and initializes its subsystems
- **Shared memory** between guests and Xen
- **PV console** by using INTC local interrupt controller + event channel

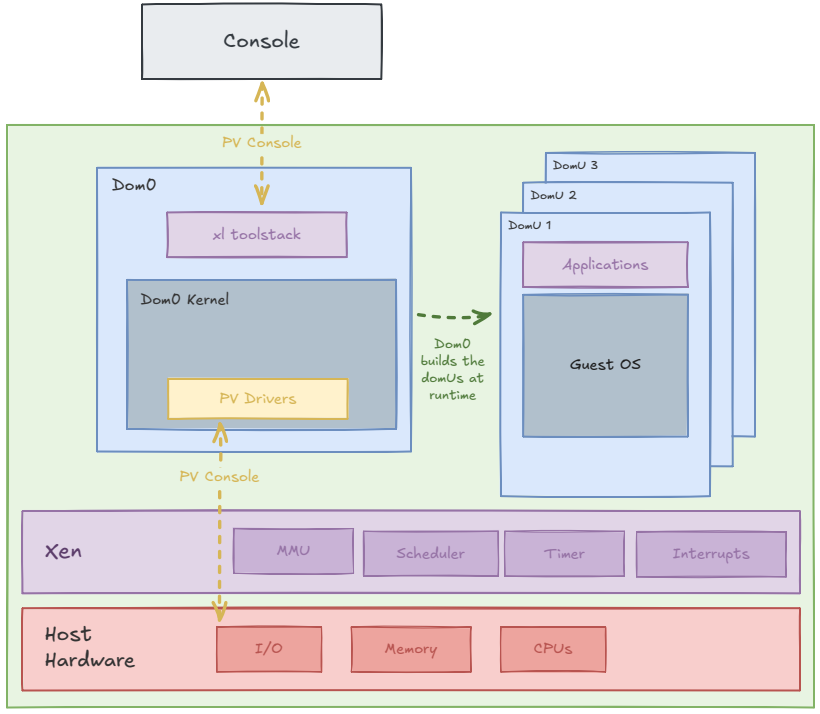


Figure 8: Xen architecture with dom0

`xl` toolstack

- `xl` = dom0 userspace program to manage guests
- First step: **getting `xl` to build** with Docker container

  → one command to build, create initrd, and launch dom0 via QEMU

Command	Status	Features required
`xl info`	Supported	Hypercalls
`xl list`	Supported	Hypercalls
`xl create`	Supported	Guest enlightenment
`xl console`	Supported	PV console support + grant tables
`xl destroy`	To implement	
other (`shutdown` , `pause` , …)	To implement	

New test framework for emulated Xen on RISC-V

Based on QTB: QEMU test bench (AMD initiative for safety-certification)

- Allows to control the QEMU emulated device from the outside (inject IRQs, read/write registers, …)
- Python based
- Architecture agnostic: can be reused to test Xen on x86/ARM

➡ **Enables a new QEMU-based Xen test strategy**

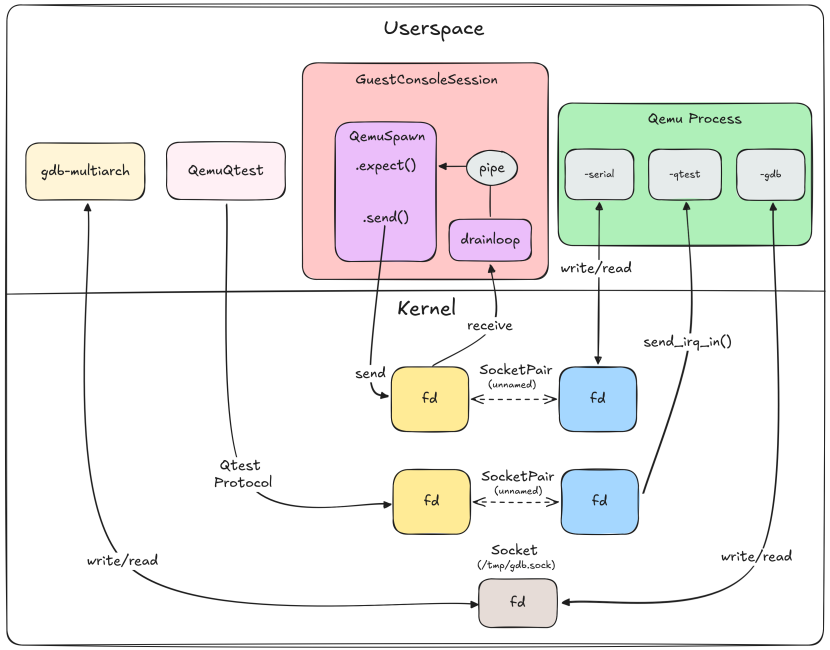


Figure 9: QEMU test bench (QTB) diagram

Part 4

Future roadmap

What's next

- `xl destroy`
- **Floating-point extension** support
- **SSTC**: guest-side support (host side already covered)
- **Newer SBI** support, beyond the legacy calls
- **IOMMU**: hypervisor-side support, for device passthrough
- **Live migration**
- **Page cache coloring**
- **Nested virtualization**
- **XTF support**: Xen Test Framework on RISC-V

Part 5

Live demo

What we'll show

`xl create` + `xl console`

Boot a domU on Xen/RISC-V via dom0 and interact with the guest via the console

`dom0less` **guest creation**


0:00 / 0:38

Questions

Thank you for your attention