

DAXFS: What Does Shared-Memory Storage Need from DAX, VFS, and CXL Memory Management?

Cong Wang, Founder and CEO, Multikernel Technologies, Inc.

Linux Plumbers Conference 2026

N Kernels, N Copies

Kernels on one machine, or hosts on one CXL pool, share memory but not caches:

- Eight kernels booting one image: eight page caches of the same rootfs
- Eight kernels serving one model: eight copies of the weights
- tmpfs, overlayfs, erofs + fscache: one cache per instance
- famfs: one metadata master
- device-dax: bytes, but no files

What we want: one namespace and one page cache, in the shared region.

DAXFS in One Slide

+-----+	+-----+	+-----+	+-----+
Super	Base image	Overlay	Page cache
4 KB	read-only	CAS hash + pool	shared slots
+-----+	+-----+	+-----+	+-----+

- **Metadata**: a hash table in the region. Insert is one `cmpxchg`
- **Writes**: copy-on-write pages; deletes are tombstones
- **Page cache**: slots go `FREE → PENDING → VALID`. The disk owner fills; others wait
- No metadata server, no log replay, no lock between participants

Multikernel and CXL: One Model

DAXFS was built for multikernel. CXL turns out to be the same model.

	Multikernel	CXL memory pool
Kernels	many, on one machine	many, on many hosts
Shared memory	carved from local RAM	CXL-attached
Who owns it	no single kernel	no single host
Coherence	one hardware domain	CXL 3.0, to be validated

Same problem, same design. Only the coherence guarantee changes.

Where It Stands

- **Validated**: many kernels on one coherent machine; shared rootfs and model weights
- **Scaffold**: CXL 3.0 multi-host, primitives behind one header, gated on real hardware
- **Target**: AI agent sandbox checkpoint and restore; a checkpoint is a file write
- **Lost at the kernel boundary**: RCU and `i_rwsem`

The filesystem works. Every question that follows is about the kernel around it.

Q1: Should fs/dax Allow Multiple Owners?

fs/dax assumes one owner per `dax_device`. So DAXFS maps pages itself, with `VM_PFNMAP`.

- **A**: extend fs/dax with a shared, multi-writer mode
- **B**: keep PFN-mapping in the filesystem; fs/dax stays single-owner
- **C**: the problem is cross-kernel PTE invalidation, and fs/dax cannot solve it

What would fs/dax need, and is it worth it?

Q2: Is a Filesystem the Right ABI?

A filesystem buys names, permissions, `mmap`, and every tool.

- **Filesystem**: everything works, and VFS assumes it owns the metadata
- **device-dax**: honest about sharing, but no names or files
- **dma-buf per object**, named by UUID as `virtio_dma_buf` does

Which abstraction should the kernel offer for pooled memory?

Is more than one acceptable?

Q3: How Does VFS Learn Another Kernel Changed Things?

Each kernel caches dentries and inodes. Today nothing invalidates them. A cached negative dentry can hide a file another kernel just created.

- **Never trust the cache:** revalidate every lookup against the region
- **Generation counters** in the region, checked on lookup
- **Explicit notification** over a cross-kernel channel

Which cost is acceptable? Is there prior art from cluster filesystems that maps here?

Q4: Per-File dma-buf over Shared Memory

Today we export the whole region, read-only. GPUs want one file. But a file is not an extent:

- Overlay pages are scattered across the pool
- A write publishes a new page
- A delete leaves a tombstone, while a device may still be reading

What are the lifetime and coherence rules for a per-file dma-buf?
Pin the file, snapshot it, or revoke the mapping?

Q5: Who Owns the Allocator Under DCD?

Today the region is fixed at creation. A CXL Dynamic Capacity Device adds and releases memory at runtime:

- Across N hosts sharing one DCD, who asks for capacity?
- What happens when a released extent still holds live overlay pages?
- **Does the filesystem grow in place, or become a set of regions?**

Q6: Memory No Kernel Owns

A spawned kernel does not own the region. No `struct page`, no LRU. Reclaim, memcg, and DAMON cannot see it.

- **Where does kernel-managed memory end and cross-kernel service memory begin?**
- **Who tracks hotness for a page no kernel owns?**
- Should memcg charge anyone for it at all?

Q7: What Does the Kernel Promise Across CXL Hosts?

Atomics are only proven inside one coherence domain. Across CXL hosts, we gate on three tests:

1. N hosts `cmpxchg` one counter with no lost updates
2. Payload published before flag is always seen before flag
3. A host crashing mid-update has defined behaviour

Is this the right gate? Should the kernel expose a primitive for it?

Who has hardware to run it?

Further Discussion

Contact: cwang@multikernel.io

Code: github.com/multikernel/daxfs

Mailing list: multikernel@lists.linux.dev