

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 can share RAM or CXL memory while keeping separate page caches:

- Buffered rootfs I/O: separate page caches in eight kernels
- Eight kernels serving one model can duplicate the weights
- tmpfs, overlayfs, erofs + fscache: caches stay local
- famfs: shared data already; centralized metadata
- 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 shared hash table with CAS-based insertion
- **Writes:** COW from base; then in-place updates
- **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: Same Model

DAXFS targets coherent sharing, CXL pooling alone is insufficient.

	Multikernel	CXL sharing
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	hardware-dependent

Cross-host atomics, failures, and memory lifetime need work.

Current Status

- **Implemented:** shared metadata and cooperative page caching
- **Validated:** Multikernel in local RAM, shared Docker image
- **Scaffold:** CXL 3.0 multi-host, core helpers exist, hardware validation pending
- **Target:** AI agent sandbox checkpoint and restore, with file-backed state

The prototype works, filesystem correctness and kernel integration remain open.

Q1: Should fs/dax Allow Multiple Owners?

DAX holders are per-kernel. DAXFS uses custom faults 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**: cross-kernel PTE invalidation needs filesystem coordination

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

Q2: Is a Filesystem the Right Interface?

A filesystem buys names, permissions, `mmap`, and tooling.

- **Filesystem**: familiar APIs, metadata needs revalidation
- **device-dax**: honest about sharing, but no names or files
- **dma-buf per object**, UUIDs as in `virtio_dma_buf`

Which abstraction should the kernel offer for pooled memory?
Is more than one acceptable?

Q3: How Does VFS Learn Another Kernel Changed Things?

DAXFS refreshes size, but lacks general cross-kernel revalidation. A cached negative dentry can hide a remote create.

- **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: the backing dma-buf, if any. Goal: per-file GPU access.

- Overlay pages may be scattered in the pool
- Writes update pages in place
- 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 Frees Memory Without an Owner?

Allocation is a cmpxchg on a shared bump pointer. Nothing frees pages today.

- A freed page may still be mapped by another kernel or its processes
- The checkpoint epoch already quiesces writers. Can it gate freeing too?
- DCD makes it worse: releasing an extent needs every page in it dead
- **Can reclamation stay lock-free, or does it need an authority?**

Q6: Can We Contain CXL Memory Failures?

Proposal: separate metadata and data regions, retire failed pages.

- How do we protect kernel copies, metadata loads, and atomics?
- How should device-dax report poison and region loss?
- How do peers retire stale pointers and PFN mappings?

What kernel support lets healthy regions keep serving files?

Q7: What Does the Kernel Promise Across CXL Hosts?

Cross-host atomics need a hardware contract, plus ordering and failure tests:

1. N hosts `cmpxchg` one counter with no lost updates
2. Seeing the flag implies seeing the payload
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