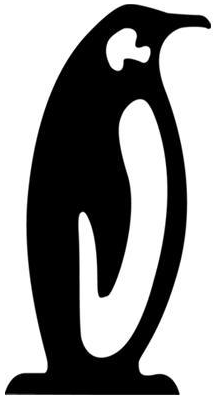


東京 2025 LINUX PLUMBERS CONFERENCE

TOKYO, JAPAN / DECEMBER 11-13, 2025

Live Update Orchestrator

The Path to Seamless Kernel Updates

Pasha Tatashin
Google

Linux Plumbers Conference 2025
Tokyo, Japan
Track: Live Update, MC
December 13, 2025

東京 2025
LINUX
PLUMBERS CONFERENCE
TOKYO, JAPAN / DEC. 11-13, 2025

What is Kernel Live Update

Definition

- Live Update is a system maintenance workflow that allows a full Kernel update without terminating running workloads.

Mechanism & Capability

- **Mechanism:** Uses kexec to boot a new kernel, bypassing the firmware/BIOS POST.
- **Capability:** Unlike standard kexec (which resets hardware), Live Update preserves the state of memory and hardware devices.

State Preservation

- **Memory:** Virtual Machine RAM is kept in place, not cleared.
- **Devices:** Ongoing operations (like DMA) continue uninterrupted during the transition.

The Result

- The new kernel "adopts" the running hardware and resumes the workload.

Live Update Architecture Stack

Kexec

- **Added in:** Kernel 2.6.13.
- **Function:** Allows the Linux kernel to boot directly into a new kernel image, bypassing the time-consuming firmware/BIOS initialization. (Follows the standard reboot path; it resets devices and does not preserve state.)

Kexec Handover (KHO)

- **Added in:** Kernel v6.16.
- **Function:** Provides the mechanisms to preserve physical memory regions (pages, vmalloc) and pass metadata (FDT) across the kexec jump.

Live Update Orchestrator (LUO)

- **Added in:** Kernel v6.19.
- **Function:**
  - **Userspace Interface**: Provides the uAPI (`/dev/liveupdate`) for the userspace agent to manage updates.
  - **Kernel Internal API:** Provides hooks for subsystems (e.g., `memfd`, `iommu`, `vfio`) to participate in the update.
  - **Resource Management:** Manages the ownership and lifecycle of preserved resources.

LUO Core Concepts

LUO File Handlers

- Kernel modules register "handlers" (e.g., memfd, vfio) to implement specific preservation logic.

LUO Sessions

- Named containers created by userspace.
- Groups file descriptors (FDs) that need preservation.
- **Lifecycle:** Tied to a session fd. If the userspace agent crashes, the session content is cleaned up automatically (prevents leaks).

Userspace Agent

- Interface: ioctl commands on /dev/liveupdate.
- Controlled via a userspace agent (e.g., luod).

LUO File Handlers

LUO uses File Handlers to support distinct resource types (memfd, vfio, etc.).

Handler API

- can_preserve(): Checks compatibility (e.g., is this FD supported by this handler).
- preserve(): Saves state; returns an opaque 8-byte handle (`serialized_data`).
  - unpreserve(): Cleans up resources if the session closes or the agent crashes before reboot.
- freeze(): Called once userspace is gone. Last chance to quiesce and check before jump.
  - unfreeze(): Rolls back state if kexec fails.
- retrieve(): Reconstructs the `struct file` in the new kernel using the serialized handle.
- can_finish(): Pre-check to ensure all dependencies are met before finish.
- finish(): Completes clean-up, and transfers ownership of the resource from LUO to userspace.

Exception State

If session finish fails or is not called, resources are held by LUO until the next reboot or LU to prevent corruption.

File-Lifecycle-Bound (FLB) State

Status Under review Patch v8.

Shared Global Resources

- Some kernel state is global but shared by multiple preserved file descriptors (e.g., IOMMU domains, HugeTLB subsystem state).
- Preserving this state individually for every file (e.g., every VFIO FD) is redundant and incorrect.
- State must be preserved once and restored once, regardless of how many files use it, and also have a defined life-cycle, when it is first created and when it is destroyed.

The Solution: FLB Objects

- A mechanism to tie the lifecycle of a global object to the aggregate lifecycle of a set of files.
- Reference Counting Logic
  - **First File Preserved:** Triggers FLB `.preserve()` callback (Serializes global state).
  - **Subsequent Files:** Increment FLB reference count.
  - **Last File Finished:** Triggers FLB `.finish()` callback (Cleans up global state).

Live Update Flow

Pre-LU	Pre-Blackout	Blackout	Post-Blackout	Post-LU
Machine is in normal state, maintenance scheduled	**Kexec** Load next kernel **LUO Agent** Creates sessions, passes them to VMM. **VMM** Calls `PRESERVE_FD` ioctl for each required resource (memfd, VFIO, etc.). **Kernel** Run `preserve()` callback for each FD.	**Userspace shutdown**: VMM suspended VMs and exits. LUO agent keeps session FDs and triggers kexec reboot call. **Kernel shutdown**: calls `freeze()` on all preserved FDs **Kernel startup:** FLB data can be accessed during boot **LUO Agent** Restores sessions, and pass them to VMMs. **VMM** Calls `RETRIEVE_FD` ioctl for each preserved resources. And performs necessary steps to resume VCPUs	**VMM** Resumes VCPUs Performs the necessary ioctls on the restored resources to perform Finish. Performs session finish, and closes session.	Machine running normally

Future Work

Userspace

- Implement Userspace agent for LUO: luod design.
- Versioning.

Extend Capabilities

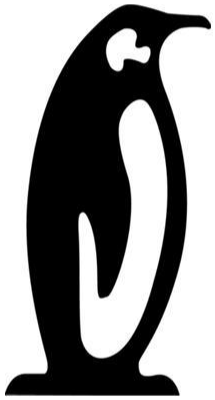
- VFIO, IOMMU, PCI, Guest_memfd, HugeTLB, TEE
- **Architectures:** Intel, AMD, and ARM64

Performance

- Deep dive is covered in Blackout Reduction talk.

Orphaned VMs

- Keep VCPUs run longer during kexec reboot.



東京 2025
LINUX PLUMBERS CONFERENCE

TOKYO, JAPAN / DECEMBER 11-13, 2025