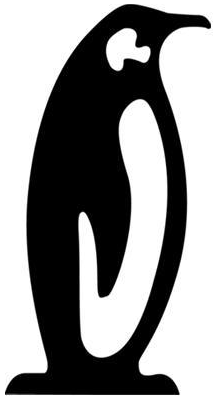


東京 2025 LINUX PLUMBERS CONFERENCE

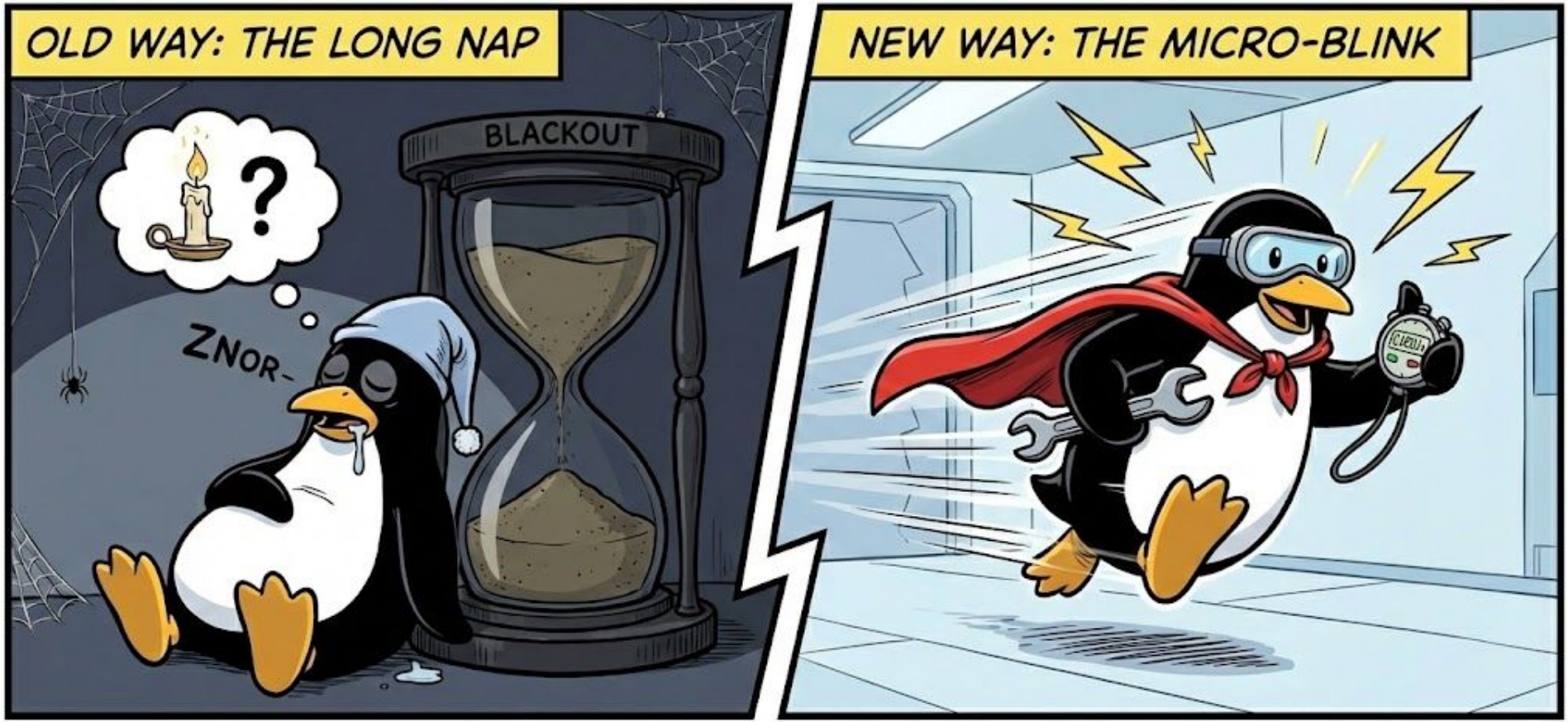
TOKYO, JAPAN / DECEMBER 11-13, 2025

Blackout Reduction

The Path to Seamless Kernel Updates

Pasha Tatashin
Google

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



From a Catnap to a Micro-Blink

Possible Optimizations, Shutdown

Device shutdown

- Take by far the most of shutdown time. There were several attempts to parallelize shutdown. Ideally, it should be simple, because during shutdown we still have other CPUs available to us, therefore there is absolutely no reason not to use other CPUs during shutdown
- [1] [PATCH v10 0/5] shut down devices asynchronously
- [2] [PATCH v6 0/3] multi-threading device shutdown

Reboot notify

- While taking less time than shutdown, in our setup, we found sleep(1) that was removed.
- Perhaps could also be parallelized?

Possible Optimizations, Purgatory and early boot

Checksum

- On x86 checksum is performed on every kexec reboot, on arm64 that is optional. Should we make it also optional on x86?

Image Relocation

- Should we allow pre-load kernel / initramfs to their final destinations like it is done in kdump? Why do we need to relocation during blackout?

Kernel Decompression

- On x86, we first go through self-extraction phase, performance depends on compression algorithm chosen.
- Why there is no uncompressed configuration?
    - Should we allow compression-none config, or pre-uncompress during kexec-load?

Possible Optimizations, Boot

"Deviceless" Hypervisor Strategy

- **Driver Removal:** Strip device drivers from the hypervisor kernel, and avoid host device initializations during blackout.
- **Device Passthrough:** Assign physical devices directly to VMs; ideally the hypervisor should manage only CPU/Memory.

I/O Elimination

- **In-Memory Only:** Enforce zero network or storage access during the blackout window.
- **State Handling:** All state preservation must occur in RAM (no disk commits).
- **Quiet Console**: No writes to console they are slow!

Kernel Configuration & Memory

- **Config Hygiene:** Remove unused kernel configs (`CONFIG_*`) to reduce boot time and binary size.
- **Deferred Initialization:** Enable `deferred struct page initialization`.
  - *Rationale:* Hypervisor memory footprint is small; Guest memory (hugepages) does not require per-base-page structs, saving initialization time, but even for hypervisor memory deferred struct page init help, unfortunately that is currently not compatible with KHO.

Telemetry

Requirements

- **Precision:** Metrics must be captured in nanoseconds, and continuous across kernels.
- **Total Blackout Metric:** from VCPU Pause to VCPU Resume is required.

Pstore for dmesg

- Since console should be silent, the pstore can be used to pass shutdown dmesg logs from current kernel to the next.

LUO Telemetry (future work**)**

1. Preserve -> freeze -> restore -> finish
2. Other breakdown can also be implemented.

Beyond Optimization - Paradigm Shifts

Live Update enables to pass resources and workload from one kernel to another, we can minimize or completely remove the downtime by either pre booting the next kernel or keeping vCPU run during reboot.

Multi-OSing / Partitioning

- Allow to pre-boot another kernel in a parallel with the hypervisor, and pass VMs to it after boot.

Devirtualization

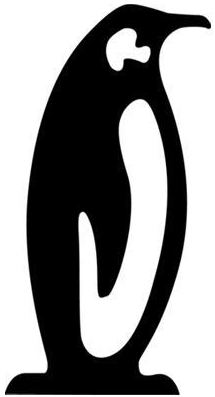
- Boot next Hypervisor into a VM, and devirtualize it.

Hibernation/Resume

- Use the hibernation to “clone” the pre-booted image of the next kernel.

Orphaned VMs

- Allow VMs to exist without VMM to allow seamless VMM update, and also to keep vcpus run longer during live update.



東京 2025 LINUX PLUMBERS CONFERENCE

TOKYO, JAPAN / DECEMBER 11-13, 2025