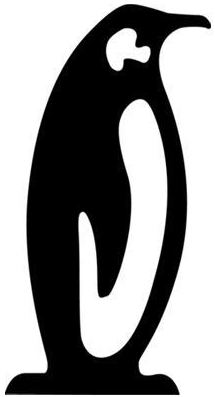


東京 2025 LINUX PLUMBERS CONFERENCE

TOKYO, JAPAN / DECEMBER 11-13, 2025

Scaling Kernel Monitoring @ Meta

Vlad Poenaru (Meta)
ヴラド ポーエナル

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

What I'm trying to achieve with this talk

Share what we've been doing

We don't have a secret sauce, but we gathered a few tools over the years. I hope the community benefits from sharing this.

Hope you all will challenge my assumptions

None of us has all the answers, and sometimes the answers we get are wrong. I'm curious how wrong we are.

See what others are doing

Continuing the previous point - I'm curious how others are solving these problems.

The Problem(s)

- **Millions of hosts:** even hard to reproduce bugs happen often enough
- **Multiple hardware types:** is it a gamma ray, a hardware errata or a kernel bug.
- **Many different services (all fail differently):** different services use the kernel differently
- **We only know 1% of the Kernel code:** no one knows everything

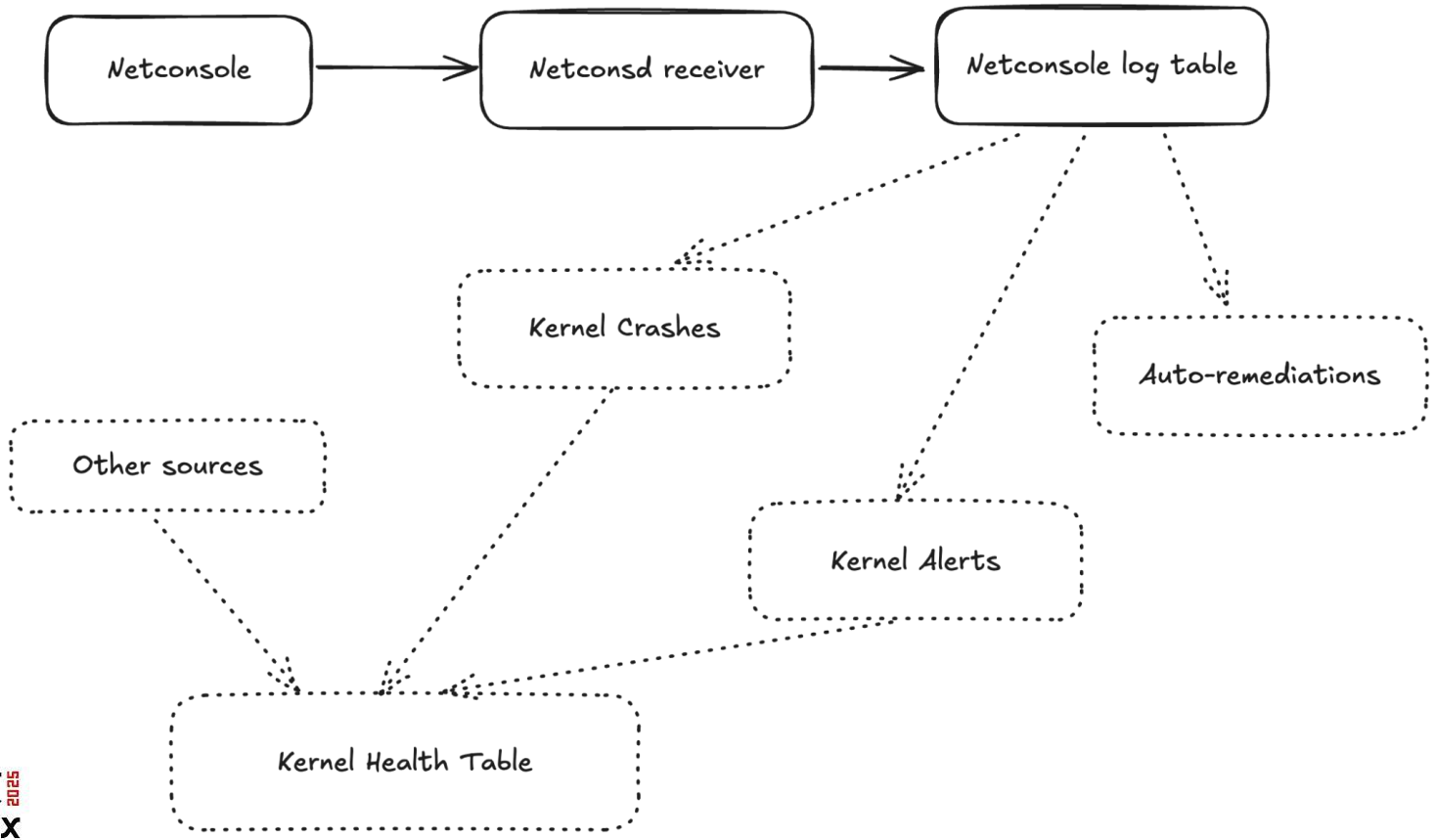
Note: This is not so much about monitoring performance.

Trivia

Meta kernel is upstream first.

Right now, running 6.16.

Preparing to start working on 6.19 next month.


Netconsole
Netconsd receiver
Netconsole log table
Kernel Crashes
Auto-remediations
Other sources
Kernel Alerts
Kernel Health Table

The tools: netconsole

Basic netcons + netconsd

Each datacenter/region runs an instance of **netconsd** listening on an anycast **VIP** which receives all **netcons** messages.

Userdata

Messages are enhanced using the **userdata** feature of netconsole - we add information like hardware model, arch, service running, kernel version, scheduler, etc.

Numbers: 1 msg/day/host

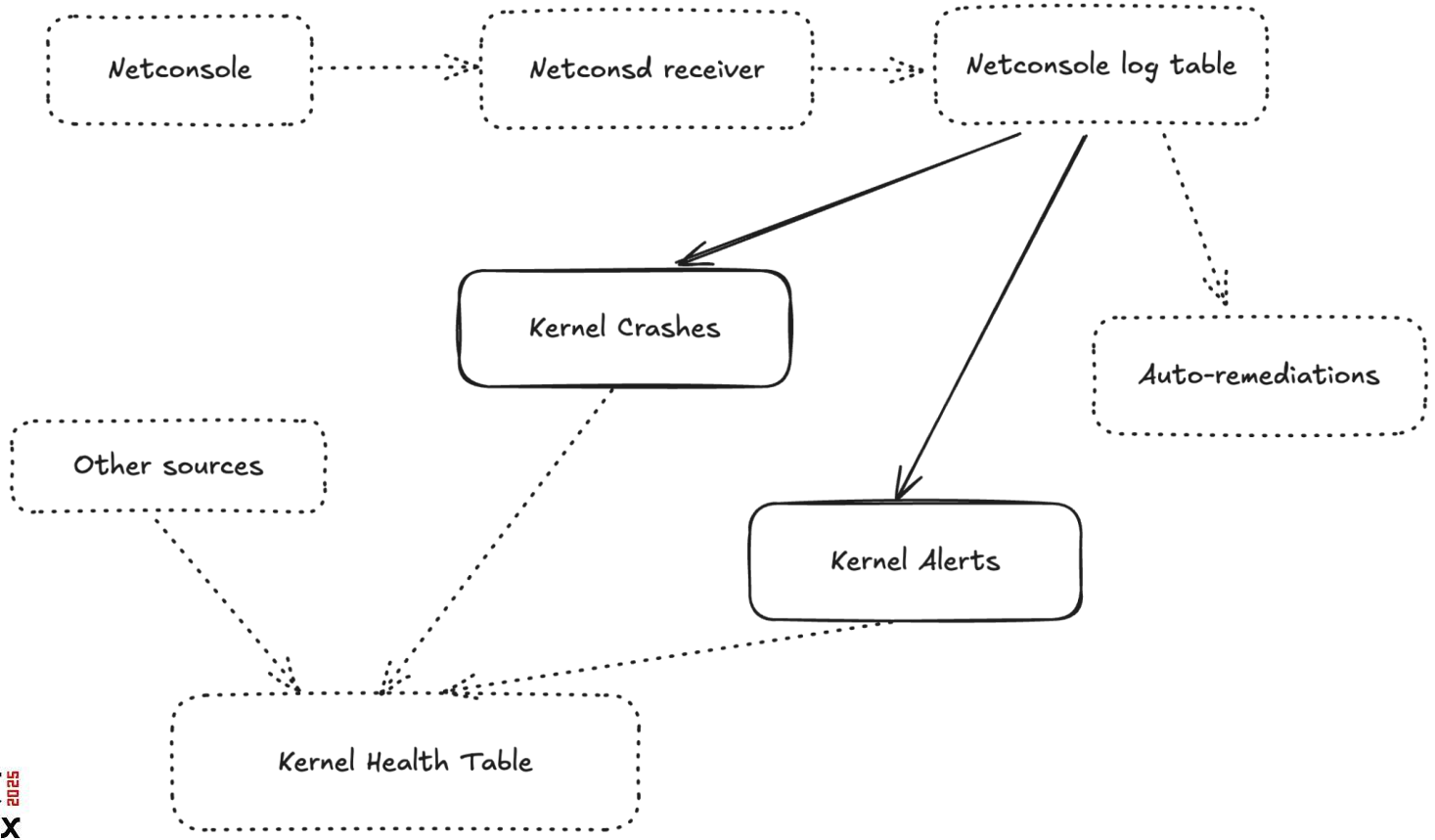
We keep 30 days worth of data "hot" while the rest goes to data warehouse.

Monitoring

Over the past year we started measuring message loss - we developed the Message ID feature.

Auto-remediations

Hosts cost is the same, even when doing nothing. Remediations based on netcons work to recover a crashed host.


Netconsole
Netconsd receiver
Netconsole log table
Kernel Crashes
Auto-remediations
Other sources
Kernel Alerts
Kernel Health Table

The tools: kdump, crash analyser

Kdump dumps the memory of the crashed kernel

Once a kernel crashes, it kexecs into a **kdump** kernel which tries to *quickly* save the RAM to local storage.

crash analyser

Next time a kernel manages to boot, we process the unprocessed dumps using a tool that orchestrates **drng, lldb,** we symbolicate and log it.

kdump + remediations = 💔

"Saving a lot of RAM on spinning disks looks like unavailability to me, let me reboot you"

Crash analyser

This is not open-sourced yet, but we hope to do it one day.

Alarms

Log post-processing

Filter noisy netcons messages, classify useful messages, group multi-line “content” - eg. stacktraces. These are not just crashes, but messages of interest - alerts, warnings, KASAN, UBSAN, etc.

Alarm Categories

Messages get normalized and classified into several alarm categories: ubsan, kasan, btrfs, oom/cgoom, firmware bugs, segfault, etc.

Alarms are not necessarily bad

We need to look at these just to make sure we're not missing something. Sometimes we need to filter the noise.

Crashes

How to find the needle in the hay*stack*?

Are two crashes alike? If they have the same stack trace, maybe yes. Hash the stacktrace and group crashes together.

Normalize counts by number of hosts

If a kernel crashes 200 times a day is it a lot? Depends on how many hosts are running that kernel. *This is still not completely solved because some crashes happen on particular hardware types, but small numbers can lie.*

Decide what "rare" means and what we don't investigate

We try to avoid investigating “rare” events, so crashes need to happen enough to be reported. Only crashes that happen more than a lower limit are automatically reported..

Triage

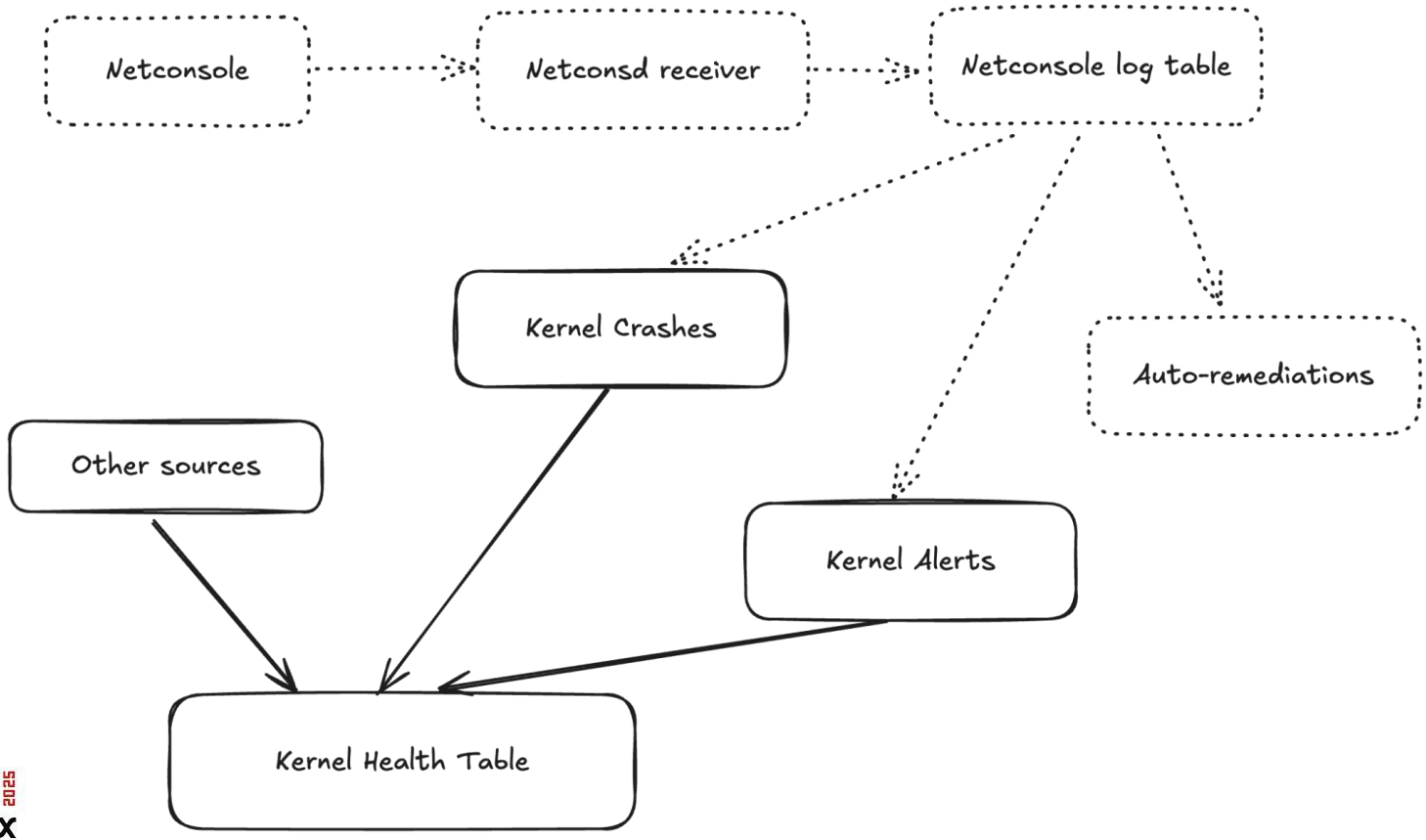
The same tool used for consolidating similar stack traces is used to triage issues. For each crash we can look at several dimensions: arch, hardware type, service running, systemd version, scheduler, etc

Logview

This is our internal tool for grouping events. It hashes the relevant bits of an event to group events that are similar.

AI

We are testing AI prompts to help analyse the crashes, look for potential fixes upstream and speed-up the investigation.


Netconsole
Netconsd receiver
Netconsole log table
Kernel Crashes
Auto-remediations
Other sources
Kernel Alerts
Kernel Health Table

Kernel Health: Rollout

How do we know if a kernel is healthy?

We look at around 10 metrics: host crashes, process crashes, eBPF failures, container failures, host connectivity, configuration management success, RCU stalls, BTRFS errors, hung tasks, CSD lock unresponsive, OOMs, hardware related errors (eg. NVME, GPU)

What do we compare with?

We came up with some static limits that work well for our fleet and dynamic metrics where the current major is compared to other majors running in the fleet.

Kernel Health Table

Denormalized table for all kernel metrics above, where dimensions are added for hardware type, other host groupings, rollout phase, kernel major

Trivia

We usually have about 4 majors running at the same time in the fleet, each with 3-4 versions.

Kernel Health is calculated for the whole fleet every 1h.

Kernel Performance

How do we know if a kernel is performing well?

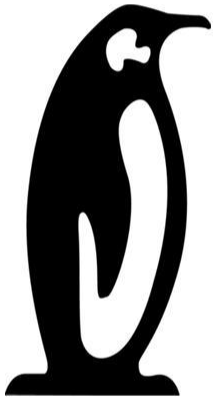
In a way this is a simpler problem - we can rely on service owners to choose their own metrics. The diversity of hardware is much smaller - a storage machine will run only run on storage services, a compute machine only on compute services.

Unsolved problems:

* When is the right time to release a kernel to a critical service?
* Should we block the release if the performance is not good enough, or should we continue and get more data?

We shouldn't get here, but ...

Questions?



東京 2025
LINUX PLUMBERS CONFERENCE

TOKYO, JAPAN / DECEMBER 11-13, 2025