PURETENSOR
INTELLIGENCE DISTILLED

The Login Daemon That Starved PID 1

A failed network block-device mount, a lingering user unit and a 1,024-reply bus limit

Paper: PT-R-2026-054 v1.0

Author: PureTensor Ltd

Date: 18 September 2026

Status: Published

Abstract

On the evening of 18 September 2026 we swapped two workstation GPUs on a multi-GPU compute node and rebooted it. The first boot came up broken in three ways at once: the desktop failed to start, SSH logins authenticated but received no login session, and 91 systemd jobs sat in the "waiting" state with none running. The obvious suspect was the hardware we had just touched. It was innocent: the kernel logged zero Xid errors (the NVIDIA driver's GPU fault reports) and the X server was healthy.

The trigger was a hung storage mount, whose own cause we could not establish. A network block-device unit mapped its first image, then hung on the ext4 mount of it, and was killed by its own 60 s start timeout before it ever mapped the second image. A lingering user manager, whose home directory depended on a mount from that second image, failed with result "dependency". The login daemon, systemd-logind 255.4, then read that unit's properties from PID 1 over D-Bus at roughly 7,300 calls a second, with no back-off, until the bus daemon reported that the daemon's connection had reached its limit of 1,024 pending replies. From that point login sessions could not be registered, and PID 1 ran none of its queued jobs, which we attribute, untested, to its job queue running at idle priority behind bus handling.

Restarting the login daemon through systemd would, on the record's reasoning, have queued behind the flood. We mapped and mounted the missing image by hand, which does not use the job queue, and sent SIGTERM to the login daemon. The job count went from 91 to 1 within 8 s. The storm had lasted 26 minutes from the first buffer-space error to the kill.

This was the second occurrence within four days (15 to 18 September) on the same machine, with a different trigger. We hardened the mapping unit that night, but the hardening has been checked only by dry run, and it removes the trigger, not the amplifier. The journal of the failing boot was not kept, so the storm figures are single observations from a record written the same night. Section 7 says which claims survive.

1. A Dead Login on One Node

The machine is the operator's primary workstation, and also a Kubernetes worker and a CI host. It runs Ubuntu with systemd 255.4 (package 255.4-1ubuntu8.17) and the D-Bus system bus daemon. A CI service account on it has lingering enabled, meaning systemd keeps that

account's per-user manager running with no one logged in; the manager hosts a rootless container daemon, and 28 CI runner services depend on that account. Ceph is the storage backend, and the node maps two RADOS block device images (RBD, Ceph's network block devices) through the rbdmap service: an image of roughly 10 TB, formatted ext4 and used for system snapshots, and a second image used for CI scratch space.

That evening two RTX PRO 6000 Blackwell cards were changed from the Max-Q edition to the Workstation Edition, with a 325 W power limit persisted. The first boot afterwards looked like a driver failure. The desktop showed a sad-face screen and then a black screen with a pointer. SSH logins succeeded at the authentication step and then failed to obtain a session. `systemctl list-jobs` showed 91 jobs, every one of them "waiting", none "running".

The question we set ourselves was whether a GPU change had caused a total loss of login, and the answer is that it had not. The kernel reported 0 Xid errors, and Xorg was healthy. What follows is the chain that was actually responsible. After recovery, Ceph reported HEALTH_OK (with two muted warnings) and the affected node was Ready; the record shows nothing about cluster state during the storm. The failing component was a single host's own init system, which no cluster-level check looks at.

One caution on evidence applies to the whole paper. The journal of the failing boot was not saved, so no raw log is available to us. Every count and timing below comes from the incident record, which the on-call agent wrote the same night, plus the git commit that carries the hardening and its message. Those two are raw artefacts. The storm itself could not be re-measured, and the table captions say which figures come from it.

2. The Night in Order

Times are British Summer Time (UTC+1). The mapping service started at 22:48:34, and the table below gives the sequence from the incident record, with durations recomputed from the recorded timestamps.

Time (BST)	Minutes after start	Event
22:48:34	0.0	rbdmap starts; maps the system-snapshot image, then hangs on its inline ext4 mount
22:49:34	1.0	rbdmap reaches its 60 s start timeout and is killed; the scratch image, listed second, was never mapped
22:49:34	1.0	The CI account's per-user manager fails with result "dependency"; all 28 runner units fail with it
22:50:35	2.0	First ENOBUFS ("No buffer space available") from the login daemon, 61 s after the dependency failure
22:50 to 23:16	2.0 to 28.0	Storm: logins fail, 91 jobs wait
23:16:36	28.0	Login daemon killed with SIGTERM (time shown is when the replacement process started); the user manager becomes active the same second
23:17:59	29.4	The graphical session target is reached after a guarded display restart
23:29:10	40.6	Hardening committed

Table 1. The boot failure and recovery. Clock times are from the incident record and the commit; the middle column is recomputed.

The storm ran 26 min 1 s from the first buffer-space error to the restart of the login daemon, and 28 min 2 s from the start of rbdmap to the same point. The unit was not slow to time out: the 60 s timeout fired on schedule, and its consequence took 26 minutes to clear.

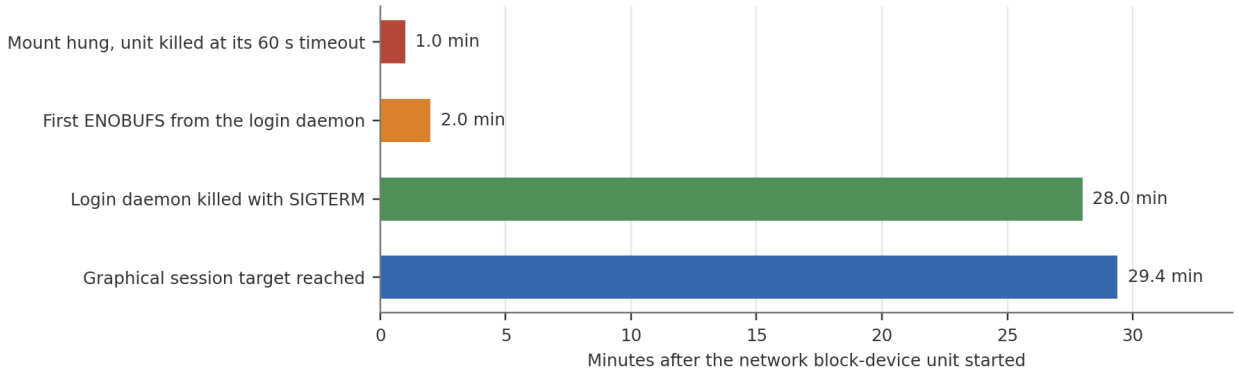


Figure 1. Minutes after the block-device unit started at which each event occurred; each bar ends at the time of the event and does not show a duration. The first two events are about a minute apart; the storm that followed lasted until the 28th minute. Values are in Table 1.

3. What We Saw During the Storm

The measurements below were taken by the on-call agent during the incident, from the incident record. None of them could be repeated afterwards.

Measure	Value
systemd-logind CPU	pinned at 100%
dbus-daemon CPU	pinned at 100%
systemd-logind resident memory	705 MB
Properties.Get calls from logind to PID 1, captured over 3 s	about 22,000 (roughly 7,300 per second)
Target of every captured call	the single per-user manager unit of the CI account
Bus daemon message	pending-reply limit reached for logind's connection, max_replies_per_connection=1024
ENOBUFS errors in the boot journal at diagnosis	60
pam_systemd result on every SSH login	"Failed to create session: No buffer space available"
systemd jobs	91, all waiting, none running
Snap-launched kubectl scope units stuck	45
GPU Xid errors	0

Table 2. Observations during the storm, each a single observation from the incident record.

Two further symptoms follow from these. The file that makes systemd refuse unprivileged logins during boot persisted, so unprivileged SSH sessions were told "System is booting up". And the desktop's session manager reported that the shell had "failed to register before timeout", fell back to a non-systemd startup path, and then could not get a session id.

The figure that characterises the storm is the call rate: one process asking PID 1 for the properties of one unit about 7,300 times a second, and the bus daemon was spending a full core on carrying the traffic.

4. How One Hung Mount Became a Bus Flood

We reconstruct the mechanism as five links. The first four rest on direct observations in the record; the fifth is the on-call agent's explanation, which fits the evidence and which we did not

test.

The first link is the storage dependency. The rbdmap service walks its image list in file order and mounts each image inline. The first image, the system snapshots, mapped; its mount hung. The unit's 60 s start timeout then killed the whole unit, so the second image, the CI scratch space, was never mapped, and nothing was mounted at the CI account's home directory (which is a bind mount of a directory on the scratch image's filesystem).

The second link is the lingering user unit. The CI account has linger enabled, and a drop-in file dated 8 September gave its user manager a RequiresMountsFor dependency on that home bind. Without the mount, the manager failed with result "dependency". Because lingering keeps the manager in play whether or not anyone logs in, the failure was not confined to a session that could be abandoned.

The third link is the retry loop, and it is the amplifier. systemd-logind 255.4 reacts to a failing per-user manager unit by reading properties of that unit from PID 1 over D-Bus in a tight loop, thousands of Properties.Get calls a second, with no back-off that we could observe in a 3 s capture. We describe the loop as observed. The on-call agent attributed it to a defect in logind and recalled similar upstream reports; we did not open those references and make no claim here about the upstream history or whether a later release fixes it.

The fourth link is the reply limit. The bus allows 1,024 pending replies per connection, and the flood saturated that allowance. What clients saw was ENOBUFS: pam_systemd could not register a login session, so SSH logins failed, and the desktop's own session lookups failed with them.

The fifth link is the starvation of PID 1. The incident record states that PID 1 serves D-Bus at normal priority but runs its job queue at idle priority, so that a bus flood leaves the queue unserviced and every transient unit start waits: login session scopes, container and snap scopes, timer-triggered services. The supporting observations are consistent with this: 91 jobs all waiting and none running, PID 1 itself low on CPU, and a queue that drained in 8 s once the flood stopped. We did not read systemd's source or trace its scheduler, so we offer this as the best-supported explanation and not a measured fact.

5. Why the Usual Fix Was Closed Off

The ordinary response to a misbehaving daemon is `systemctl restart systemd-logind`. Here that command was itself a job, and jobs were the thing PID 1 was not running. On the record's reasoning, it would have joined the 91 and waited behind them. The incident record gives the recovery as four steps that avoid the queue or follow once it is moving.

Step	Action	Why it works
1	Map the scratch image by hand, mount it, mount the CI home bind	mount(8) does not go through the job queue; it supplies the missing dependency
2	SIGTERM to the login daemon (it has Restart=always, RestartSec=0)	A signal does not go through the queue either; the flood stops and PID 1 drains
3	Guarded display restart	Verdict OK: the pre-check passed; session on seat0 active and shell registered
4	Start what the storm blocked	28 runner units, rbdmap, one OCR service; clear 45 failed kubectl scopes

Table 3. Recovery in order. The job queue is bypassed by steps 1 and 2.

After the signal, jobs fell from 91 to 1 within 8 s. We write "91 to 1" because that is what the record says; the one remaining job was not counted as stuck. The login daemon, configured to restart immediately, came back as a new process, and the per-user manager became active in

the same second as the kill, because the mount it needed was now present.

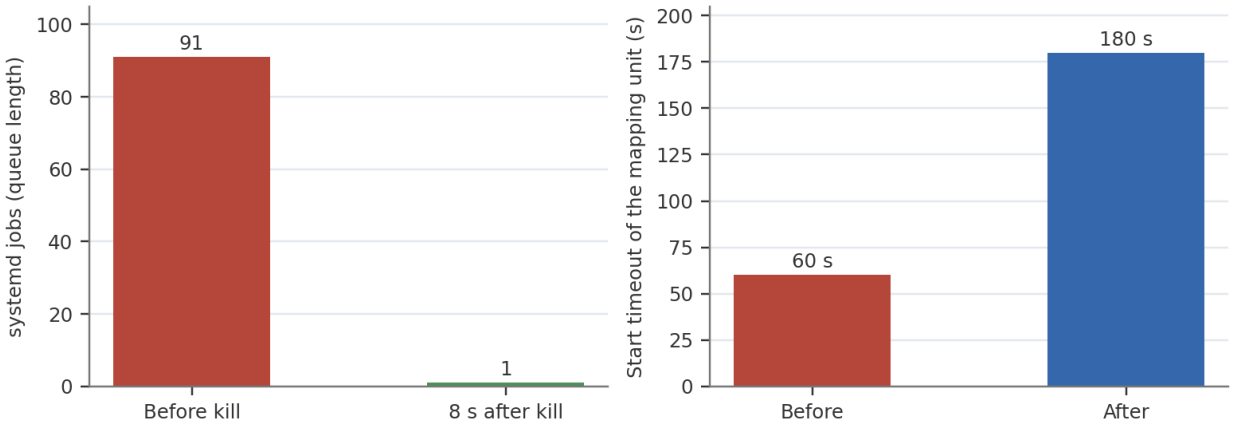


Figure 2. Left: systemd jobs waiting before the login daemon was killed and 8 s after. Right: the start timeout of the block-device mapping unit before and after the hardening. The values 91 and 1 are given in Section 5; 60 s and 180 s are in Table 4.

One detail in step 4 is worth recording. We started the 28 runner units by glob pattern, and the first attempt did nothing, because a glob start only matches units that are already loaded in memory. We then enumerated the units from their files on disk and started them by name.

At the end of the session the checks were as follows. SSH created a real session, the boot-time login refusal file was gone, and there had been 0 ENOBUFS errors since 23:16:40. The 28 runners were active, and 8 of 8 GPU inference services were active. Ceph reported HEALTH_OK, with two muted authentication warnings, with 24 of 24 OSDs up, three monitors in quorum, and 785 placement groups active+clean (two scrubbing, one deep scrubbing), with 34 TiB used of 183 TiB.

6. The Hardening, and What It Leaves Open

We committed a drop-in for the mapping unit at 23:29:10 and installed it live. It changes the unit's behaviour as set out below.

Property	Before	After
TimeoutStartSec	60 s	180 s
TimeoutStopSec	default (90 s)	90 s, set explicitly
Start order	Stock map of images in file order, scratch second; scratch mounts only in a post-start step	Step 1: map scratch (60 s), settle udev, mount scratch (60 s), mount CI home bind (20 s), each guarded by an already-mapped or already-mounted check. Step 2: stock map for the rest, failure-tolerant, 90 s
Snapshot image mount	Post-start step with no timeout of its own (errors ignored)	Failure-tolerant ('-' prefix), 30 s timeout
Stop	Unmap everything	Lazily unmount the snapshot tool's run-directory mounts and the snapshot mount first, then unmap everything
Image order in the map file	Snapshot image first	Scratch image first

Table 4. The rbdmap drop-in, before and after.

The design principle is that the mount the CI account's home depends on is established first, under its own short timeouts, and cannot be starved by the slower and less important snapshot

image. A hung snapshot mount may still hang, but it can no longer prevent the scratch image from being mapped and mounted first. We also deleted a stale five-minute "wait for monitors" drop-in that existed in the repository but had never been installed.

The tests were a dry run of step 1, which exited 0 in 16 ms because everything was already mapped, a stop-side dry run that listed exactly the two snapshot mounts, and a clean `systemd-analyze verify`. The real test is the next reboot, which had not happened at the time of writing. The hardening is therefore verified only by dry run.

Two things are not fixed. The trigger is removed, but the amplifier is not: the incident record states that other failures of that unit could reach the same amplifier. A detector for that condition was filed as a follow-up task, and nothing had shipped by the time of writing. We also do not know why the snapshot-image mount hung. The record's account is that the previous shutdown logged an unmap failure for the first image, because the snapshot tool still held a mount of it under its run directory, and that stale state made the next boot's inline mount hang. That is an inference from shutdown logs that no longer exist; the hang itself was not traced. The stop-side change in Table 4 targets this hypothesis, which is a reason for caution about reading it as confirmed.

7. What These Numbers Will Not Carry

This is one incident on one host in one boot. There was no repeat and no controlled reproduction. No raw log of the storm was kept. The 91 jobs, the 8 s drain, the 22,000 calls in 3 s and the 60 buffer-space errors are each a single observation, recorded by the on-call agent that night, and we cannot put an error bar on any of them. The 8 s is the figure in the same-night session report; a separate memory note rounds it to about 10 s, and we use the former.

The causal step from the reply limit to the starvation of PID 1 rests on correlated observations and an explanation we did not test. The rest of the chain, from the hung mount to the failed manager, is supported by the timestamps in Table 1: each event follows the previous one by the interval you would expect. The boot also followed hardware work, which is a confound for any reader inclined to blame the cards; the zero Xid count and a healthy X server rule the GPUs out, and the recovery did not touch them.

The claims that survive are these: the dependency chain from mount to lingering manager; a flood of about 7,300 property reads a second; the bus daemon's message about the 1,024-reply limit; recovery within 8 s of killing the login daemon; and the content of the hardening. The claims that do not survive as stated are that the GPUs or the display stack were at fault, and that restarting the login daemon through systemd would have worked.

Four questions remain open. We do not know why the snapshot mount hung. We have not checked whether the retry loop is a known defect already fixed upstream. We have not verified the split between normal-priority bus handling and idle-priority job running in PID 1's source. And we do not know whether the hardening survives a real reboot.

A fifth is a discrepancy in our own records. A boot-simplification policy set on 15 September listed rbdmap as disabled and CI linger as off, yet both were active on the 18th. The repository history shows a boot-safe drop-in existed from 15 September and rbdmap ran on the 18th, so the policy was not in force that night. We do not know when it was reverted.

Finally, this was not the first time. Three days earlier, on 15 September, a reboot on the same machine failed in the same class: an RBD map failure (there, an authentication key-type problem after a key change), a dependent mount failing, a lingering user service failing, and the login daemon looping until the same 1,024-reply limit was reached and every login failed. The root cause was different, and the earlier incident was resolved by issuing new keys. The mount-to-logind-retry-to-bus-exhaustion sequence had already been written down as a trap that

day; what the 15th lacked was the job-queue starvation account above. A different trigger reached the same amplifier twice within four days (15 and 18 September), which is consistent with the view that fixing triggers one at a time is not enough, though two incidents cannot show it. We observed only one package build (255.4-1ubuntu8.17, Ubuntu-patched), and Ubuntu patches may differ from upstream.

8. What We Changed in Practice

The immediate change is the drop-in in Table 4, installed live, with scratch-first ordering, a longer start timeout, a stop path that releases the snapshot mounts before unmapping, an explicit stop timeout, and failure tolerance for the slow image. The next experiment is the plain one: reboot, with the mapping unit's journal captured and kept, and confirm that both images are mapped and mounted within the new timeouts and that the lingering manager comes up. If it does, we will have verified the hardening; if the snapshot mount hangs again, we will have the stack and kernel messages that we lacked on 18 September. In either case we will keep the journal. The detector for this condition remains to be built, and until it is, the amplifier is still in place.

9. Methods and Artefacts

The system was a single multi-GPU compute node (two of its Blackwell cards were changed that evening) running Ubuntu with systemd 255.4 (package 255.4-1ubuntu8.17) and the D-Bus system bus daemon (version not recorded in the incident record and not verified by us). The Ceph release was not recorded. The tools used during the incident were journalctl, loginctl, a busctl monitor on the system bus, `systemctl list-jobs`, the rbd device map command, mount(8) and kill. There are no statistics in this paper: every figure is a single observation. We re-read the commit and its drop-in diff, recomputed durations from the recorded timestamps and found no raw log to re-measure. What is retained is the contemporaneous session report and memory note, the git commit with the full drop-in diff, and the earlier repository history of the drop-ins; the failing boot's journal, the bus monitor capture and the process listing were not kept.

PureTensor operates a sovereign AI infrastructure fleet (on-premises GPU compute, storage, and serving) run day-to-day by autonomous agents under human direction. This paper is PT-R-2026-054. The measurements come from a single boot on 18 September 2026, recorded the same night; the commit and the incident record are retained, and the failing boot's journal was not kept.