Lusankya (Network Attached Storage)

UnRAID Server in a QNAP Box

System Identity & Hardware

Overview

Lusankya is the QNAP TS-873 chassis, repurposed from stock QNAP QTS to bare-metal **UnRAID 7.2.4** (previous version: 7.2.3) after the original QNAP OS failed. This page documents the hardware as captured in diagnostics on 2026-06-23.

Hardware

Component	Detail
Chassis	QNAP TS-873 (8-bay)
Motherboard	AMD Bettong CRB, AMI BIOS QY03AR53 (2018-03-01, BIOS rev 5.12)
CPU	AMD Embedded R-Series RX-421ND, 4 cores/4 threads, 1.4–2.1 GHz
RAM	31 GiB total (no swap configured)
Boot device	Internal flash (`sda`, 30GB partition)

Network

Interface	Role	Address
`br0` (eth0)	Primary LAN bridge	`192.168.1.77/24`
`eth2`	2.5GbE NIC	`192.168.1.119/24`
`tailscale1`	Remote access overlay	`100.70.97.16/32`

- Server hostname resolves as `Lusankya.local` — no SSL/HTTPS configured on the management UI (HTTP only, port 80).
- DNS rebinding protection is **disabled**, so `myunraid.net` remote-access URLs will resolve on this LAN.

Notes / Gotchas

- This hardware previously ran QNAP QTS; the OS migration to UnRAID is a known discontinuity point — any documentation or scripts referencing the old QTS Container Station predate this rebuild and are stale.
- No swap configured — if RAM pressure becomes an issue under heavier Docker workloads, there's no overflow cushion.

Last Updated

2026-06-23 (from diagnostics snapshot `lusankya-diagnostics-20260623-1011`)

Array Configuration & Shares

Overview

Lusankya's array uses **UnRAID's single-parity protection model** — this is *not* traditional RAID 6. One dedicated parity drive protects against a single data-disk failure; there is no second parity drive in this configuration. (`mdNumDisks=8` → 1 parity + 7 data disks.)

> **Correction worth flagging:** earlier references to this as an "8-bay RAID 6" setup are inaccurate to how UnRAID actually protects data. Each data disk is independently formatted XFS — UnRAID computes parity across disks rather than striping like classic RAID. Functionally it tolerates one failed disk (or two, only if a second parity drive is added), but the mechanism and recovery process differ meaningfully from hardware RAID 6.

Array Layout

Role	Device	Filesystem
Parity	sdg	— (parity, no FS)
disk1–disk6	sdh–sdm	XFS
disk7	sdn	XFS
Cache pool	sde + sdf	BTRFS, RAID1 (2× 256GB SSD)

Cache Pool

- **Profile:** BTRFS RAID1, 2 devices (mirrored) — `Timetec 35PN2280SATA-256GB` × 2
- Hosts `docker`, `appdata`, and `system` shares exclusively (`shareUseCache="only"`)
- Docker image: `/mnt/cache/docker/docker.img`, 20GB, native backing FS
- Custom Docker networks defined: `eth1 eth3 eth4`

Shares

Share	Cache Use	Disks
`appdata`	Cache only	cache pool
`docker`	Cache only	cache pool
`system`	Cache only	cache pool
`Data`	No (array direct)	disk1, disk2, disk6, disk7
`Photos`	No (array direct)	disk1, disk2, disk5, disk6, disk7
`Multimedia`	No (array direct)	disk1–disk7 (all)

All three array shares export NFS as `private` security with an explicit host allow rule for `192.168.1.85` (Centerpoint) only.

USB Backup Targets (Unassigned Devices)

These are **not array members** — they're Unassigned Devices plugin-managed USB drives, matching the rclone backup scripts already documented for this host:

Mount	Drive	Raw Capacity	Used
`/mnt/disks/Expansion24`	ST24000DM001	24TB	13TB / 22TB usable (58%)
`/mnt/disks/Expansion26`	ST26000DM000	26TB	19TB / 24TB usable (80%)

Notes / Gotchas

- `appdata` / `docker` / `system` are cache-only — if the cache pool fills or both SSDs fail simultaneously, Docker and VM services go down hard since Mover has nothing to fall back to for those shares.
- Array capacity is getting tight: disk1–disk5 are sitting at 87–94% utilized. disk7 (the newest member, WD Red 5TB) is the slack — only 30% used. Worth planning a rebalance or disk7 expansion before disk1–disk5 fill completely.

Last Updated

2026-06-23

Disk Inventory & SMART Health

Overview

Full physical disk inventory from the 2026-06-23 diagnostics SMART reports. Age is derived from `Power_On_Hours`, which reflects cumulative powered-on time, not calendar age since purchase — a drive bought 3 years ago but spun down often will show fewer hours than its shelf age.

Array Disks

Role	Model	Serial	Capacity	Power-On Hours	Approx. Age	Reallocated Sectors	Pending Sectors	SMART Health
Parity	ST6000VN0033-2EE110	ZAD9HJ1G	6TB	56,390	~6.4 yrs	0	0	PASSED
disk1	ST6000VN0033-2EE110	ZAD9CS4N	6TB	56,390	~6.4 yrs	0	0	PASSED
disk2	ST6000VN0033-2EE110	ZAD9HMSB	6TB	56,390	~6.4 yrs	0	0	PASSED
disk3	ST6000VN0033-2EE110	ZAD9HPXB	6TB	56,391	~6.4 yrs	0	0	PASSED
disk4	ST6000VN0033-2EE110	ZAD9HM6T	6TB	56,391	~6.4 yrs	0	0	PASSED
disk5	ST6000VN0033-2EE110	ZAD9HN0N	6TB	56,390	~6.4 yrs	0	0	PASSED
disk6	ST6000VN0033-2EE110	ZAD9HP9K	6TB	56,390	~6.4 yrs	0	0	PASSED

Role	Model	Serial	Capacity	Power-On Hours	Approx. Age	Reallocated Sectors	Pending Sectors	SMART Health
disk7	WDC WD50EFRX-68MYMN1	WD-WX11DC4498K3	5TB	46,143	~5.3 yrs	0	0	PASSED

Cache Pool

Role	Model	Serial	Capacity	Power-On Hours	Approx. Age	SMART Health
cache	Timetec 35PN2280SATA-256GB	QY250626A2C0721	256GB	5,665	~0.65 yrs	PASSED
cache2	Timetec 35PN2280SATA-256GB	QY250626A2C0719	256GB	5,545	~0.63 yrs	PASSED

Unassigned USB Backup Drives

Mount	Model	Serial	Capacity	Power-On Hours	Approx. Age	SMART Health
Expansion24	ST24000DM001-3Y7103	ZXA0VGZY	24TB	6,647	~0.76 yrs	PASSED
Expansion26	ST26000DM000-3Y8103	ZXA0DK4L	26TB	3,898	~0.44 yrs	PASSED

Boot/Unmonitored Devices

Device	Role	Note
sda	Internal boot flash	No SMART attributes reported — normal for USB flash media; not a fault
sdb	USB module	No SMART attributes reported — same as above

Fleet Summary

> ✅ **Zero reallocated sectors, zero pending sectors, zero offline-uncorrectable sectors across every spinning and solid-state disk in the fleet.** Every drive reports SMART overall-health PASSED. This is a clean bill of health as of this snapshot.

The 7× Seagate IronWolf array drives are the oldest hardware at ~6.4 years powered-on time — they're the ones to watch first for any future SMART degradation, given they're well past the 5-year mark commonly cited as when AFR (annualized failure rate) starts climbing for NAS-class drives.

Notes / Gotchas

- Power-on hours ≠ calendar age. If these IronWolf drives were purchased used or pulled from another array, actual unit age could exceed 6.4 years.
- Re-run diagnostics quarterly and diff this table — a sudden jump in Reallocated_Sector_Ct or Current_Pending_Sector between snapshots is the earliest warning sign, well before SMART flips to FAILED.

Last Updated

2026-06-23

Parity Checks & Mover

Overview

UnRAID maintenance centers on two automatic background processes: **parity checks** (data integrity verification) and **Mover** (cache-to-array file migration). Neither is configured to run on a schedule in the current diagnostics snapshot — both should be reviewed.

Parity Checks

- A parity check reads every data disk simultaneously, recomputes parity, and compares it against the parity drive — it confirms the array is internally consistent, **not** that the data itself is uncorrupted.
- **Non-correcting** checks (report-only) are the safe default for routine scheduling; **correcting** checks should only run after an unclean shutdown or to fix errors a non-correcting check already found.
- UnRAID automatically triggers a parity check after any unsafe shutdown — using a UPS is the primary defense against triggering this unnecessarily.
- Recommended cadence for this array size (6TB drives): **monthly**, scheduled during low-usage overnight hours via Settings → Scheduler → Parity Check. A full check on a 6TB array typically runs 4–12 hours depending on concurrent load.

Mover

- Mover transfers files between cache and array based on each share's Primary/Secondary storage and cache-use settings.
- On this host, `appdata`/`docker`/`system` are cache-only (Mover takes no action on them) — `Data`/`Photos`/`Multimedia` write directly to the array (Mover also has nothing to move for these, since they bypass cache entirely).
- Since none of the array-facing shares use cache as a write buffer, Mover schedule is largely inactive on this host today — worth confirming this is intentional rather than an oversight, since it means large incoming writes to those shares go straight to spinning disk at HDD write speed rather than bursting to SSD first.

Notes / Gotchas

- **Always disable Docker and VM Manager before manually running Mover** — open file handles from running containers can cause incomplete or failed moves.
- Mover activity is also the trigger for the NFS stale-file-handle issue documented on the Gotchas page — when Mover relocates a file, its underlying fileid changes, and NFS clients holding the old handle break.

Last Updated

2026-06-23

USB Cold-Backup Scripts (Expansion24 / Expansion26)

Overview

Two UnRAID User Scripts mirror array shares to two USB-attached cold-backup drives via `rclone sync`. Both scripts were extended with Pushover push notifications on completion (success or failure), since User Scripts has no built-in alerting.

Backup Mapping

Target Drive	Shares Covered
Expansion24 (24TB)	`Data` (full), `Photos`, `Multimedia/Movies`, `Multimedia/ST`
Expansion26 (26TB)	`Multimedia/Audio`, `Multimedia/Books`, `Multimedia/Personal`, `Multimedia/TV`, `Multimedia/To Review`, `Multimedia/Ubooquity`, `Multimedia/audiobookshelf`

Script Logic (both jobs share this pattern)

```
#!/bin/bash

# === Pushover Config ===
PUSHOVER_TOKEN="<redacted — see Pushover application dashboard>"
PUSHOVER_USER="<redacted — user key, not stored in script comments>"

FAILED_JOBS=""
START_TIME=$(date +%s)
```

```
send_pushover() {
  local STATUS="$1"
  local MESSAGE="$2"
  curl -s \
    --form-string "token=${PUSHOVER_TOKEN}" \
    --form-string "user=${PUSHOVER_USER}" \
    --form-string "title=UnRAID Backup - <ExpansionXX>" \
    --form-string "message=${MESSAGE}" \
    --form-string "priority=$( [ "$STATUS" = "FAIL" ] && echo 1 || echo 0 )" \
    https://api.pushover.net/1/messages.json > /dev/null
}

run_sync() {
  local SRC="$1" DEST="$2" LOG="$3" LABEL="$4"
  rclone sync "$SRC" "$DEST" --log-file="$LOG" -v
  if [ $? -ne 0 ]; then
    FAILED_JOBS="${FAILED_JOBS}\n- ${LABEL}"
  fi
}

# One run_sync call per share/subfolder pair...

END_TIME=$(date +%s)
DURATION=$(( (END_TIME - START_TIME) / 60 ))

if [ -z "$FAILED_JOBS" ]; then
  send_pushover "OK" "All jobs completed successfully in ${DURATION} min."
else
  send_pushover "FAIL" "Backup finished with failures in ${DURATION} min.\nFailed
jobs:${FAILED_JOBS}"
fi
```

Key design points:

- Each `rclone sync` call is wrapped in `run_sync()`, capturing its own exit code — a failure in job 2 of 7 doesn't mask or get overwritten by job 7's exit code.
- All jobs run independently to completion regardless of earlier failures (no `set -e`) — by design, so one broken share doesn't block backups of everything else.
- A single Pushover notification fires at the end summarizing success or listing exactly which job(s) failed, with failure notifications set to `priority=1` to bypass Pushover quiet hours.

Notes / Gotchas

- Both `PUSHOVER_TOKEN` and `PUSHOVER_USER` should be redacted from any wiki copy of this script — store actual credentials in the User Scripts plugin directly, not in a doc that might get exported or shared.
- `$?` only reflects the immediately preceding command — if a future edit pipes `rclone` output through another command (`| tee`, `| grep`, etc.), the exit code capture breaks silently. Keep the exit-code check directly after the `rclone` call.
- These backups are **not parity-protected** — they live on Unassigned Devices, outside the array. A single backup drive failure has no recovery path other than re-running the sync from source.

Last Updated

2026-06-23

UnRAID Operational Gotchas

Overview

Accumulated friction points from running UnRAID on Lusankya and integrating it with Centerpoint's Docker stack. Each entry includes root cause and the working fix.

1. NFS "Stale File Handle" After Mover Runs

Symptom: A client (Centerpoint) holding an NFS mount to a Lusankya share suddenly gets `Stale file handle` on `ls` or file access, requiring an unmount/remount to recover.

Root cause: When Mover relocates a file between cache and array, the underlying FUSE file ID changes. NFS clients cache file handles tied to the old fileid; UnRAID's `fuse_remember` tunable controls how long those handles are cached client-side, and a mismatch after a Mover-triggered fileid change produces "stale handle" until the client refreshes.

Fix:

- Force NFS **v4.2** on the client mount (NFSv3 is far more prone to this than v4.x).
- In `/etc/fstab` on the client, use `_netdev,nofail,x-systemd.automount,x-systemd.idle-timeout=300` rather than a static mount — automount re-establishes the handle on next access instead of hanging on a dead one.
- On the UnRAID side, the `fuse_remember` tunable (Settings → Global Share Settings) can be tuned if this happens frequently outside Mover events — but automount on the client side resolves it without touching server config.

2. Centerpoint Mounts Not Loading After Restart

Symptom: After a reboot, none of Centerpoint's NFS/CIFS mounts to Lusankya come back automatically; Docker containers depending on them fail to start.

Root cause: Docker services start before network mounts are ready at boot — a race condition, not a Lusankya-side fault.

Fix: `sudo mount -a` recovers immediately. To prevent recurrence, add `_netdev` to the relevant `/etc/fstab` lines so systemd waits for network availability before attempting the mount, and consider `x-systemd.automount` for more graceful boot-time handling.

3. Portainer 500 Error on Compose Redeploy After Changing NFS Path

Symptom: Updating a stack's NFS source path (new IP or mount point) in Portainer fails with a generic `Request failed with error code 500`.

Root cause: Docker refuses to redefine an existing **named volume's** `driver_opts` in place — the old volume definition with the stale NFS config is still registered.

Fix: Either `docker volume rm` the stale named volumes before redeploying, or — the more durable fix — replace named NFS volumes with direct bind mounts to an already-working host-level mount (`/mnt/data/...`) per the established Centerpoint convention of direct bind mounts over named volumes with `driver_opts`.

4. Immich / Any App Expecting Sentinel Files on a Pre-Existing Library Path

Symptom: An app (e.g. Immich) crashes on startup when its data volume points at a pre-existing directory rather than an empty one — it's looking for first-run sentinel files (`.immich`, etc.) that only get created during fresh initialization.

Fix: Manually create the expected subdirectories and sentinel files before first container start, with ownership matching the container's expected UID (commonly 1000).

5. UnRAID Array ≠ Traditional RAID 6

Clarification, not a bug: UnRAID's array uses per-disk XFS/BTRFS filesystems with a dedicated parity calculation layer — single parity drive here means tolerance for **one** failed data disk, recovered by rebuilding from the remaining disks + parity. This is functionally different from striped RAID 6 (which tolerates two failures by design). Don't assume RAID-6-equivalent fault tolerance when planning around this array; a second parity drive would be required to match that.

6. Cache-Only Shares Have No Mover Fallback

Symptom: None directly observed yet on this host, but worth flagging — `appdata`/`docker`/`system` are configured `shareUseCache="only"`. If the 2-disk BTRFS cache pool fills or both SSDs fail, there's no array fallback; writes simply fail.

Mitigation: Monitor cache pool free space proactively; don't let it run consistently above ~85% utilized.

7. Boot Flash / USB Enclosures Don't Report Standard SMART

Symptom: Diagnostics SMART reports for the boot flash drive and any USB-bridged drive come back empty — easy to misread as a tool failure.

Root cause: Most USB-to-SATA bridges and flash media don't pass through SMART attributes the same way native SATA/SAS does. This is expected, not a fault — but it does mean those devices have **no early-warning health signal**, so physical inspection / replacement-on-schedule is the only mitigation for the boot flash itself.

Last Updated

2026-06-23