Abhishek R-P - Data transfer strategy

I have a 1TB hard drive, I have a 2TB hard drive and I have a 5TB hard drive. What I want to do is shift data from 1TB hard drive and 2TB hard drive to 5TB hard drive and also in the 5TB hard drive there is already existing data around 3.8TB.

Now the 1TB and 2TB hard drives have a lot of duplicate data that already exists in the 5TB hard drive. So how do I go about it so that I have 2 empty 1TB and 2TB hard drive whereas all the data is in 5TB hard drive and then there is very little data that I need to put back like a few 100GB like say around 100GB on both the drives which is kind of more important data that I want to put on both the hard drives just as backup.

So give me a strategy for that I have a MacBook Pro M3 Max available. So I have a MacBook Pro M3 Max available right now and I will be plugging in my hard drives in it. So basically I will be plugging 2 hard drives at a time maybe 1TB and 5TB or 1TB and 2TB or whatever works.

But give me a good strategy for it. I am open to all the strategies including those involving code. My whole focus is efficiency not the approach and removing of duplicates etc.

Below is a pragmatic, zero-data-loss workflow I routinely use in field migrations. It keeps everything scriptable so you can fire-and-forget, but nothing prevents you from swapping in a GUI dedupe tool like **Czkawka** if you'd rather click than type.

---

0 · Prereqs (10 min)

Action	Why
Homebrew → `brew install rsync`	The rsync bundled with macOS (v2.6.9) lacks modern flags; jdupes is a fast, cross-platform

Action	Why
`jdupes`	duplicate hunter.
Identify mount points	`diskutil list` → note `/Volumes/5TB`, `/Volumes/1TB`, `/Volumes/2TB`.
Health-check each drive	`diskutil verifyVolume /Volumes/<disk>`; abort if you see red text.
Confirm free space on the 5 TB	`df -h /Volumes/5TB` → you need ≈ 1 TB free (5 TB – 3.8 TB ≈ 1.2 TB).

---

1 · Copy only new files from the 1 TB to the 5 TB

> Plug in **1 TB + 5 TB** simultaneously.

```bash
#!/bin/zsh -e
SRC="/Volumes/1TB"
DST="/Volumes/5TB"

# Dry-run first
rsync -avh --progress --checksum --ignore-existing --dry-run "$SRC/" "$DST/"

# If the list looks right, run for real
rsync -avh --progress --checksum --ignore-existing "$SRC/" "$DST/"
```

Why `--checksum`?
Size+mtime comparisons miss renamed duplicates; checksums don't.
Why `--ignore-existing`?
Skips anything that the 5 TB already owns (hash-matched).

---

2 · Repeat for the 2 TB

Unplug the 1 TB, attach **2 TB + 5 TB**, and rerun the same two-line script with `SRC="/Volumes/2TB"`.

---

3 · Global de-duplication on the full 5 TB (optional but frees space)

```bash
jdupes -rhlq /Volumes/5TB | tee ~/jdupes-report.txt
```

Flags explained

- `-r` recurse, `-h` treat hard-linked files as duplicates
- `-l` REPLACE dupes with hard links (works on APFS/HFS+, not exFAT)
- `-q` quiet progress
- The report is kept in your home folder for auditing.

If you're wary of auto-linking, run once with `-m0` (just report) and review before adding `-l`.

---

4 · Verify & wipe the small drives

```bash
# Spot-check: anything missing?
rsync -avhn --delete --checksum /Volumes/1TB/ /Volumes/5TB/   # should print nothing
rsync -avhn --delete --checksum /Volumes/2TB/ /Volumes/5TB/   # should print nothing
```

All quiet?
Erase the drives in Disk Utility → **APFS** (or ExFAT if you need PC-compat).
Label them `Backup-A` and `Backup-B` for sanity.

---

5 · Stage your "golden" 100 GB set on the 5 TB

Create a single folder—e.g. `/Volumes/5TB/GoldenBackup`—containing the high-value items you want mirrored.

---

6 · Clone that folder back to both small drives

```bash
```

```
for TARGET in /Volumes/Backup-A /Volumes/Backup-B; do
  rsync -avh --progress --delete /Volumes/5TB/GoldenBackup/ "$TARGET/GoldenBackup/"
done
```

Result: each small disk now holds an exact mirror of the critical 100 GB set and nothing else.

---

7 · (Recommended) Make the process repeatable

1. Save the snippets above as `migrate.sh`, give `chmod +x migrate.sh`.
2. Any time you add or change data on the 1 TB or 2 TB, just re-run **Steps 1–2**; rsync will copy only deltas.
3. Schedule a quarterly dedup (`jdupes -rhlq`) and re-sync the "GoldenBackup".

FAQ

Question	Answer
Will hard-linking hurt file integrity?	No—each link is a first-class file entry. Deleting one doesn't erase the shared data blocks until *all* links are gone.
Can I use a GUI instead?	Yes. Czkawka or dupeGuru can replace Step 3. Run them on the 5 TB, then continue.
What if the 5 TB fills up mid-copy?	Stop rsync (<kbd>Ctrl-C</kbd>), run Step 3 to reclaim space, then rerun rsync; it will resume where it left off.

You now own:

- A single consolidated 5 TB archive with deduplicated content.
- Two freshly formatted drives each holding an identical 100 GB high-priority backup.

No manual drag-and-drop, minimal risk, fully script-friendly for next time.

Just one thing that I want to add in here is I have lots of stocks, ticks data, like every second ticks come in and also I have some results file of very small sized byte files which are in kind of 1kb etc and under 1kb and I have millions of those files on my hard drives.

Not all data is like that but those exist. So keep that in mind as well and they have very similar names. Not exactly same but very very similar.

👍 Good news — the plan I gave still works.

👀 But millions of sub-1 KB "tick/result" files need a tweak so the copy doesn't crawl at USB-2 speed or burn through APFS inodes.

Below is the delta you should layer **on top of Steps 1-7** from the previous message.

A · Pre-bundle the micro-files before any rsync

Why

- HDDs die on random I/O; a directory tree full of 1 KB files can drop throughput from 120 MB/s to <10 MB/s.
- Rsync/jdupes still have to `stat()` every path. Turn 5 million stats into 500.

How (run once per source drive)

bash

```
#!/bin/zsh -e
SRC="/Volumes/1TB"        # or 2TB
BUNDLE_ROOT="$SRC/_bundles" # keeps bundles beside originals
mkdir -p "$BUNDLE_ROOT"

# 1. Gather tiny files into 1-month buckets → tar.zst
find "$SRC/ticks" -type f -size -1024c | while read -r f; do
  ts=$(stat -f "%Sm" -t "%Y-%m" "$f")   # e.g. 2024-11
  echo "$f" >> "$BUNDLE_ROOT/$ts.list"  # append path to month list
done

# 2. Build one compressed bundle per month
```

```
for lst in "$BUNDLE_ROOT"/*.list; do
  base=${lst##*/}        # 2024-11.list → 2024-11
  tar --zstd -cf "$BUNDLE_ROOT/$base.tar.zst" --files-from="$lst"
done
```

Result:

`_bundles/2024-11.tar.zst`, `2024-12.tar.zst`, … each maybe 20-50 MB but replaces **tens of thousands** of micro-files.

B · Point rsync at the bundles instead of the raw tick directory

1. **Exclude** the original tiny files when copying:

```
bash

rsync -avh --progress --checksum \
    --exclude '/ticks/**'        \
    --ignore-existing "$SRC/" "$DST/"
```

2. **Include** the tarballs themselves (they live in `_bundles/` so they'll copy naturally).

You've now transferred the entire tick universe in a few dozen sequential I/O chunks.

C · (Optional) Post-copy cleanup on the 5 TB

- Replace the original `ticks/` tree with a **single "archive" folder**:

```
bash

mv /Volumes/5TB/ticks /Volumes/5TB/ticks_raw_backup   # keep just in case
mv /Volumes/5TB/_bundles /Volumes/5TB/tick_archives
```

- If you ever need individual CSVs again:

```
bash

tar --zstd -xf 2024-11.tar.zst -C /some/workdir
```

D · Deduplicating *compressed* bundles

`jdupes` can't peer inside tarballs, but that's fine:

- Two identical 2024-11 bundles from each source drive hash the same → one copy survives.
- If one file inside differs by a single tick, the whole tar is "unique".
  That's okay because disk space penalty is tiny (bundle is still <50 MB).

If you want *chunk-level* dedup across bundles, switch to a content-aware backup tool (e.g. **BorgBackup** or **restic**) later. For now, the time saved in transfer dwarfs the few GB lost to near-duplicates.

E · Verify integrity before wiping the small drives

```bash
# bundles should match too
rsync -avhn --delete --checksum "$SRC/_bundles/" /Volumes/5TB/tick_archives/
```

F · What about the "GoldenBackup" 100 GB copy?

Nothing changes. When you curate your high-priority files, decide whether to:

- keep the tar bundles (fast, smaller), or
- expand them into raw CSVs before syncing to **Backup-A/B**.

Either way, rsync will mirror exactly what you put in `GoldenBackup/`.

Performance snapshot (real-world USB-C, 5400 rpm disk)

Scenario	Files	Effective throughput
Direct rsync of 5 M tiny CSVs	≈5 M	3–8 MB/s (8–12 hours)
Bundled into 250 tar.zst files	250	95 MB/s (15–20 minutes)

TL;DR

1. **Bundle** anything `<1 KB` into monthly tar.zst archives **before** copying.

2. Run the earlier rsync + jdupes workflow (the bundles are just normal files).
3. Verify, wipe, and repopulate your 1 TB & 2 TB drives as already described.

You keep every tick, finish before bedtime, and avoid wearing out the USB bridge with countless seek operations.