FIELD NOTES FROM A PRESALES SOLUTIONS ARCHITECT

Sizing Object Storage Honestly — Raw × 0.6 Is What You Can Sell

A presales field note about the one number that decides an object storage design — the gap between raw disk and sellable capacity — and the checks I now run before I put a terabyte figure in any proposal.

Daanial Farrukh · 09 November 2026 · https://quabin.com/object-storage-sizing-raw-capacity-sellable/

The request is always short.

"We need 100 TB of S3."

It comes from an application team. Backups, documents, file sync, media — the workload changes, the sentence does not.

The first draft of the design usually answers it just as quickly. Four servers. Disks that add up to a round 100 TB. A line in the proposal that says "100 TB object storage".

I ask one question before anything goes out.

"After parity, and after we stop filling at a safe level — how much of that 100 TB can the customer actually write?"

Silence. Then someone opens a calculator.

The honest answer is about 60 TB.

That's the trap. Raw capacity does not show up as a problem on day one. The cluster comes up, the dashboard shows a big number, the first uploads land. The gap only appears months later, when the bucket is two thirds full and the "100 TB" system is already out of room.

What "Sellable Capacity" Actually Means

Sellable capacity is the amount of data a customer can store while the cluster still survives a failure and still has room to heal. It is not the sum of the disk labels.

Two things sit between the two numbers.

The first is **erasure coding**. Object storage like MinIO does not use RAID. It splits each object into data shards and parity shards and spreads them across drives and servers. Parity is what lets you lose drives — or a whole server — without losing data. It is also capacity you paid for and cannot sell.

A simple example: an erasure set of 12 drives with 3 parity drives keeps data on 9 of 12. That is 75% usable.

The second is **the fill limit**. A cluster at 100% cannot rebalance, cannot absorb a growth spike and cannot heal cleanly after a failure. I plan to stop at 80%.

Put together:

Raw disk → × 0.75 (after parity) → × 0.8 (fill limit) → ≈ raw × 0.6 sellable

So the sizing formula I use is plain:

sellable ≈ raw × 0.6

And the reverse, which is the one that matters in a proposal:

raw ≈ sellable ÷ 0.6

You see this gap in three places most often:

- Backup targets, where retention grows quietly every month.
- File sync and share platforms that put user content on S3.
- Media and archive buckets, where objects are large and nobody deletes anything.

Why Nobody Sees It in Testing

A proof of concept runs on a few hundred gigabytes. It never gets near the fill limit. Parity is invisible because every drive is healthy.

The console also does not help. Most dashboards show total and used, and the total is raw or near-raw. Nothing on that screen says "you planned to stop at 80%".

So the number that goes into the proposal is the number the dashboard shows. And the number the dashboard shows is the wrong one to sell.

The Worked Example: 100 TB Sellable

Here is the same request, sized honestly.

100 TB sellable ÷ 0.6 ≈ 167 TB raw.

The layout I would propose:

- 4 servers, each a 12-bay chassis.
- 6 × 8 TB HDD in each server to start — 24 drives, 192 TB raw.
- 6 empty bays per server for growth.

192 TB raw × 0.6 ≈ 115 TB sellable. That covers the 100 TB with a little headroom, and the empty bays mean the next step is adding disks, not buying a new platform.

Here's the part that trips people up: **when you grow, add drives in equal counts and equal sizes to all nodes.** Six more 8 TB drives in one server and none in the others gives you an unbalanced cluster, not more usable space.

The Hardware Rules Behind the Number

The 0.6 only holds if the hardware is built the way object storage expects. These are the rules I hold to.

Commodity x86 servers with local disks, not a SAN. Object storage scales out. You add servers, not shelves to one array. A SAN puts all the data behind one controller pair — the exact single point the software was designed to remove.

HDD for the capacity tier. Object I/O is mostly whole-object, sequential reads and writes. HDD handles that well, and price per TB decides most of these deals. Use NVMe for the OS and metadata only. I switch the capacity tier to SSD only when the average object is smaller than about 1 MB and latency really matters.

No RAID. Give the software raw disks through an HBA in IT mode. Erasure coding already does the redundancy. RAID under it means you pay for parity twice and still hide failed disks from the software that is supposed to heal them.

Four nodes minimum. Four is the smallest count that survives losing a whole server and still lets you patch one node at a time. With three, one planned reboot plus one surprise failure takes the service down. An SLA on fewer than four nodes is not honest.

Check the licence before you sell it. MinIO is AGPL, with a commercial licence on the side. If you offer it as a service, read the terms with your legal team before the proposal goes out, not after.

The Second Trap: Not Everything Lives in the Bucket

Capacity is half of "honest". The other half is what the bucket actually protects.

I proved this in a lab with a file sync platform (oCIS) using its S3 driver, STORAGE_USERS_DRIVER: s3ng. I uploaded a 192 KiB file. It appeared in the MinIO bucket as a UUID-named object, deep under the space ID.

That part worked as designed. But the file names, the folder tree, the shares and the permissions did not go to the bucket. They stayed on the application server's local disk, under /var/lib/ocis.

So the flow looks like this:

User upload → oCIS → file content → S3 bucket User upload → oCIS → names, shares, permissions → local disk

Replicate only the bucket to a second site and you have every byte of content — and no way to know which file is which.

DR for this kind of platform is two jobs: replicate the object store, and separately protect the small metadata store. The metadata is small, so it costs little. It is also the part people forget.

One more detail from the same lab: the bucket must exist before oCIS starts. Plan it in the build order.

How to Actually Check This

These checks work on a running MinIO cluster. Replace myminio with your alias. All of them are read-only.

Step 1: See the real raw and used numbers.

```
mc admin info myminio
```

This lists every server, every drive and its state, and the used and total capacity. Count the drives per server. They should match across all nodes. If one node shows more drives than the others, growth was not done evenly.

Step 2: Check the parity you are actually running.

```
mc admin config get myminio storage_class
```

This shows the storage class settings, including the parity used for standard objects. The 0.75 in the formula assumes a specific ratio — for example 3 parity drives in a 12-drive set. If your deployment uses more parity, usable space is lower, and your sellable number must come down with it. Do the arithmetic with your real data-to-total ratio, not mine.

Step 3: Confirm the disks are raw, not RAID volumes.

```
lsblk
```

Run it on each server. You want to see each data drive as its own device, mounted on its own path. One large device built from many disks means a RAID controller is in the way.

Step 4: Check replication is healthy.

```
mc admin replicate status myminio
```

This is MinIO site replication, set up with mc admin replicate add. It copies objects, IAM users and policies between sites, asynchronously, usually with seconds of lag. The status should show every site and no growing backlog.

Step 5: The verification step.

This step is where most tutorials stop and where mine starts.

Site replication only copies changes made after it was set up. If the first site already held data, the second site does not have it. Also, every site except the first must be empty when you add it.

So after setup, run:

```
mc admin replicate resync start myminio myminio-dr
```

Then compare object counts per bucket on both sides.

```
mc ls --recursive --summarize myminio/bucket-name
```

```
mc ls --recursive --summarize myminio-dr/bucket-name
```

The totals at the bottom should match, give or take objects still in flight. If the second site shows far fewer objects, that is not a bug. It is the verification — the old data was never copied.

Plain takeaway: **replication that started late is a backup of the last few weeks, not of the platform.**

The Alternatives I Rejected

Customers often suggest simpler designs. These are the ones I say no to, and why.

1. **One shared volume across two data centres.** The data still lives in one place, so it is not DR. And two database instances writing to one data directory will corrupt it.
2. **rsync for a live database.** It copies file by file over minutes. The copy of an InnoDB database at the end is not consistent. Use the database's own replication.
3. **rsync in a loop as "continuous protection".** Every run rescans the whole tree. That is a batch job on a timer, not continuous. For block-level continuous protection use a CDP product; for file-level near-real-time, use inotify-based tools such as lsyncd.

The pattern I propose instead: native database replication, object storage site replication, and one site active at a time.

The Checklist

Before any object storage figure goes into a proposal:

1. Raw = sellable ÷ 0.6. Check: write both numbers in the proposal, not just one.
2. Parity matches the formula. Check: mc admin config get myminio storage_class.
3. Four or more nodes. Check: mc admin info myminio lists at least four servers.
4. No RAID under the drives. Check: lsblk on each server.
5. Equal drives on every node. Check: drive count per server in mc admin info myminio.
6. Old data reached the DR site. Check: mc ls --recursive --summarize on both sites.
7. Metadata outside the bucket is protected. Check: find where the application keeps names and permissions, and confirm it has its own backup or replication.

What I Tell Every Customer Now

In every object storage review, I ask one question:

"At 80% full, how many terabytes have you stored — and is that the number in your contract?"

If the answer is yes, they're in good shape. If the answer is "the contract says the disk size", we're not done yet.

I write about the questions I ask in real customer rooms — the ones that change architectures and surface the problems no one else noticed. If you found this useful, drop your email below and I'll send the next one when it's ready.

FAQs

How much usable space does object storage give from raw disk?

Plan for about 60% of raw as sellable capacity. Erasure coding parity takes roughly a quarter in a common layout, and a fill limit of 80% takes another fifth. Your exact figure depends on the parity ratio your deployment actually uses.

Should I use RAID under MinIO?

No. Give MinIO raw disks through an HBA in IT mode and let erasure coding handle redundancy. RAID underneath means paying for parity twice and hiding failed drives from the software that heals them.

Why is four nodes the minimum for production object storage?

Four servers is the smallest count that survives the loss of a whole server and still lets you patch nodes one at a time. With fewer, a planned reboot plus one failure can take the service down, so an availability promise is not honest.

HDD or SSD for object storage?

HDD for the capacity tier in most cases, because object I/O is mostly whole-object and sequential and price per TB matters most. Keep NVMe for the OS and metadata. Consider SSD for capacity only when the average object is under about 1 MB and latency is critical.

Does MinIO site replication copy existing data?

No. Site replication copies changes made after it is set up, and every site except the first must be empty when added. Run mc admin replicate resync start to copy the existing objects, then compare object counts on both sides.

Is replicating the S3 bucket enough for disaster recovery?

Not always. Some applications keep file content in S3 but names, folders, shares and permissions on local disk. Without that metadata, the replicated objects cannot be mapped back to files, so protect it separately.

Read online: https://quabin.com/object-storage-sizing-raw-capacity-sellable/