Understanding RAID: How performance scales from one disk to eight

Enlarge / Behold—96TB of storage stacked on a workbench in an unwieldy, eight-high spiral. Don't try this at home, kids; photography and system administration don't mix very well.
Jim Salter

reader comments

33 with 18 posters participating, including story author

Share this story

One of the first big challenges neophyte sysadmins and data hoarding enthusiasts face is how to store more than a single disk worth of data. The short—and traditional—answer here is RAID (a Redundant Array of Inexpensive Disks), but even then there are many different RAID topologies to choose from.

Most people who implement RAID expect to get extra performance, as well as extra storage, out of all those disks. Those expectations aren't always rooted very firmly in the real world, unfortunately. But since we're all home with time for some technical projects, we hope to shed some light on how to plan for storage performance—not just the total number of gibibytes (GB) you can cram into an array.

A quick note here: Although readers will be interested in the raw numbers, we urge a stronger focus on *how* they relate to one another. All of our charts relate the performance of RAID arrays at sizes from two to eight disks to the performance of a single disk. If you change the model of disk, your raw

numbers will change accordingly—but the relation to a single disk's performance will not for the most part.

Equipment as tested

- Yes, I work in a largely unfinished basement. At least I've got windows out into the yard. Don't @ me.
  Jim Salter
- This is the Summer 2019 Storage Hot Rod, with all twelve bays loaded and hot. The first four are my own stuff; the last eight are the devices under test today. (The machine above it is banshee, my Ryzen 7 3700X workstation, in an identical 12-bay chassis.)
  Jim Salter

We used the eight empty bays in our Summer 2019 Storage Hot Rod for this test. It's got oodles of RAM and more than enough CPU horsepower to chew through these storage tests without breaking a sweat.

The Storage Hot Rod's also got a dedicated LSI-9300-8i Host Bus Adapter (HBA) which isn't used for anything but the disks under test. The first four bays of the chassis have our own backup data on them—but they were idle during all tests here, and are attached to the motherboard's SATA controller, entirely isolated from our test arrays.

How we tested

As always, we used fio to perform all of our storage tests. We ran them locally on the Hot Rod, and we used three basic random-access test types: read, write, and sync write. Each of the tests was run with both 4K and 1M blocksizes, and I ran the tests both with a single process and iodepth=1, and with eight processes with iodepth=8.

For all tests, we're using Linux kernel RAID, as implemented

in the Linux kernel version 4.15, along with the ext4 filesystem. We used the --assume-clean parameter when creating our RAID arrays in order to avoid overwriting every block of the array, and we used -E lazy_itable_init=0,lazy_journal_init=0 when creating the ext4 filesystem to avoid contaminating our tests with ongoing background writes initializing the filesystem in the background.

Kernel RAID vs hardware RAID

We do not have side-by-side tests with a hardware RAID adapter here, so you'll need to take our word for it when we tell you that hardware RAID is not magic. We *have* privately tested Linux kernel RAID versus popular professional, dedicated eight-port hardware RAID cards several times over the years, however.

For the most part, kernel RAID significantly outperforms hardware RAID. This is due in part to vastly more active development and maintenance in the Linux kernel than you'll find in firmware for the cards. It's also worth noting that a typical modern server has *tremendously* faster CPU and more RAM available to it than a hardware RAID controller does.

The one exception to this rule is that some hardware RAID controllers have a battery-backed cache. These cards commit sync write requests to the onboard, battery-backed cache instead of to disk, and they lie to the operating system about it. Cached, synchronous writes aggregate then trickle out from the controller's cache to disk. This works—and performs—just like asynchronous writes that are aggregated and committed by the operating system itself.

Asynchronous writes greatly outperform synchronous writes, and so this represents a significant boost to such a controller's performance. The card relies on the battery to ensure survival of the cached data across power outages. This is, for the most

part, like putting the entire server on a UPS and using the amusingly yet appropriately named libeatmydata, which causes the operating system to lie to *itself* about the result of fsync calls.

A word to the wise: If the battery fails in a RAID controller and the controller does not detect it, corruption can and will result after power outages, since the card is still lying to the operating system and applications when they request assurances that data has been committed safely to disk. If the controller *does* proactively detect the battery failure, it simply disables on-card write aggregation entirely—which returns sync write performance to its true, far lower level.

In our experience, administrators are overwhelmingly likely not to notice when a hardware controller's cache batteries fail. Frequently, those administrators will still be operating their systems at reduced performance and reliability levels for years afterward.

One final warning about hardware RAID controllers: It's difficult to predict whether a hardware RAID array created under one controller will import successfully to a different model of controller later. Even if the model of controller remains the same, the user interfaces of the management applications or BIOS/UEFI routines used to import arrays are frequently written in *incredibly* unclear language.

We find that with hardware RAID, it's frequently difficult to tell whether you're nuking your array or importing it safely. So in the event of a controller failure and replacement, you may end up sweating bullets, YOLOing, and hoping. *Caveat imperator.*

https://arstechnica.com/?p=1659718