fio Disk I/O benchmark

CONCEPTS

> Writing to a raw device (/dev/sdX, /dev/nvme0n1) destroys its data. Benchmark a file on the filesystem, or a scratch device you can wipe.

Job file / CLI workload
↓ parsed
Jobs numjobs clones
↓ submit
ioengine iodepth in flight
↓ complete
Report IOPS, BW, latency
One fio run

RUNNING

fio *<jobfile>*
Run a job file

fio --name=*<job>* **--rw=***<mode>* **...**
Define a job on the CLI

fio --section=*<name>* *<jobfile>*
Run only one section

fio --output=*<file>* **--output-format=***<fmt>*
Write report; normal/terse/json/json+

fio --minimal *<jobfile>*
Terse semicolon-separated output

fio --showcmd *<jobfile>*
Convert job file to CLI

RUNNING (cont.)

fio --parse-only *<jobfile>*
Check options, no I/O

fio --readonly ...
Refuse write jobs

fio --cmdhelp=*<option>*
Help for one option

fio --enghelp
List ioengines

fio --eta-newline=*<time>*
One progress line per interval

fio --version
Show version

ACCESS PATTERN

--rw=read / write
Sequential read / write

--rw=randread / randwrite
Random read / write

--rw=randrw --rwmixread=*<pct>*
Mixed random I/O

--rw=rw
Mixed sequential I/O

--bs=*<size>*
Block size, e.g. 4k or 1m

--bs=*<r>*,*<w>*
Separate read,write sizes

--bssplit=*<size>*/*<pct>*:...
Weighted block sizes

SIZE AND FILES

--filename=*<path>*
File or device to test

--directory=*<dir>*
Place generated files here

--size=*<size>*
I/O region per job

--offset=*<size>*
Start offset in target

--nrfiles=*<n>*
Files per job

--unlink=1
Delete files when done

ENGINE AND QUEUE

--ioengine=libaio
Linux async I/O

--ioengine=io_uring
Modern Linux async I/O

--ioengine=psync
Plain pread/pwrite (default)

--ioengine=sync
Basic read/write

--iodepth=*<n>*
I/Os kept in flight

--direct=1
O_DIRECT, skip page cache

--sync=1
Open with O_SYNC

--fsync=*<n>*
fsync every N writes

ENGINE AND QUEUE (cont.)

--end_fsync=1
fsync at job end

TIME AND LOAD

--runtime=*<time>*
Stop after this long

--time_based
Loop until runtime ends

--ramp_time=*<time>*
Warm-up excluded from stats

--numjobs=*<n>*
Parallel clones of the job

--group_reporting
One summary for all clones

--rate=*<bw>*
Cap bandwidth, e.g. 10m

--rate_iops=*<n>*
Cap IOPS

JOB FILES

Job file layout

```
[global]
ioengine=libaio
direct=1
runtime=60
time_based

[randread-4k]
rw=randread
bs=4k
iodepth=32
```

[global]
Defaults for all jobs

JOB FILES (cont.)

stonewall
Wait for earlier jobs

OUTPUT AND LOGS

--output-format=json
JSON report

--write_lat_log=*<prefix>*
Per-I/O latency log

--write_bw_log=*<prefix>*
Bandwidth log

--write_iops_log=*<prefix>*
IOPS log

--log_avg_msec=*<ms>*
Average log entries

--verify=crc32c --do_verify=1
Check data read back

RECIPES

4k random read IOPS

```
fio --name=randread --
filename=<file> --size=2g \
  --rw=randread --bs=4k --
ioengine=libaio --iodepth=32
\
  --direct=1 --runtime=60 --
time_based --group_reporting
```

Sequential write throughput

```
fio --name=seqwrite --
filename=<file> --size=4g \
  --rw=write --bs=1m --
ioengine=libaio --iodepth=8
\
  --direct=1 --end_fsync=1
```