Benchmarking

fio

Description

> Flexible IO Tester (Fio) is a benchmarking and workload simulation tool for Linux/Unix created by Jens Axboe, who also maintains the block layer of the Linux kernel. Fio is highly tunable and widely used for storage performance benchmarking.
> ...
> There are also ways to run Fio on Windows, but generally other tools that are better suited for Windows OS, such as IOmeter or CrystalDiskMark are recommended.

Performance benchmarking with Fio on Nutanix

Usage

Fio Argument	Description
`--name=str`	Fio will create a file with the specified name to run the test. The file will be created at the specified path or in the current working directory if only a short name is provided.
`--ioengine=str`	Defines how the job issues I/O to the test file. Key engines include `libaio` (Linux native), `solarisaio` (Solaris native), `posixaio` (POSIX), `windowsaio` (Windows native), and `nfs` (asynchronous I/O to NFS).
`--size=int`	The size of the file on which Fio will run the benchmarking test.
`--rw=str`	Specifies the I/O pattern. Options include `read`, `write`, `randread`, `randwrite`, `rw`, and `randrw`. Fio defaults to a 50/50 read/write mix for `rw` and `randrw`, adjustable with `--rwmixread`.

Fio Argument	Description
`--bs=int`	Defines the block size for the I/O generation. Default is 4k, but it is recommended to specify the block size explicitly for more accurate testing.
`--direct=bool`	Use `true=1` for non-buffered I/O, which bypasses the OS filesystem cache. This setting is recommended for fair testing.
`--numjobs=int`	Number of threads spawned by the test. Use `--group_reporting` to aggregate results across threads.
`--iodepth=int`	Number of I/O units to keep in flight against the file, representing the amount of outstanding I/O per thread.
`--runtime=int`	Specifies the duration (in seconds) for which the test will run.
`--time_based`	If set, the test will run for the specified `runtime`, repeating the workload as many times as possible within that duration.
`--startdelay`	Adds a delay (in seconds) between the test file creation and the actual test.
`--sync`	Forces Fio to use synchronized I/O. This ensures that I/O operations are completed before proceeding, which can be used to simulate workloads that require strict data consistency.

Test Description	Fio Command
Sequential writes with 1Mb block size. Imitates write backup activity or large file copies.	`fio --name=fiotest --filename=test1 --size=4Gb --rw=write --bs=1M --direct=1 --numjobs=8 --ioengine=libaio --iodepth=8 --group_reporting --runtime=30 --startdelay=60`
Sequential reads with 1Mb block size. Imitates read backup activity or large file copies.	`fio --name=fiotest --filename=test1 --size=4Gb --rw=read --bs=1M --direct=1 --numjobs=8 --ioengine=libaio --iodepth=8 --group_reporting --runtime=30 --startdelay=60`
Random writes with 64Kb block size. Medium block size workload for writes.	`fio --name=fiotest --filename=test1 --size=4Gb --rw=randwrite --bs=64k --direct=1 --numjobs=8 --ioengine=libaio --iodepth=16 --group_reporting --runtime=30 --startdelay=60`
Random reads with 64Kb block size. Medium block size workload for reads.	`fio --name=fiotest --filename=test1 --size=4Gb --rw=randread --bs=64k --direct=1 --numjobs=8 --ioengine=libaio --iodepth=16 --group_reporting --runtime=30 --startdelay=60`
Random writes with 8Kb block size. Common database workload simulation for writes.	`fio --name=fiotest --filename=test1 --size=4Gb --rw=randwrite --bs=8k --direct=1 --numjobs=8 --ioengine=libaio --iodepth=32 --group_reporting --runtime=30 --startdelay=60`
Random reads with 8Kb block size. Common database workload simulation for reads.	`fio --name=fiotest --filename=test1 --size=4Gb --rw=randread --bs=8k --direct=1 --numjobs=8 --ioengine=libaio --iodepth=32 --group_reporting --runtime=30 --startdelay=60`

Additional Reading

- fio.readthedocs.io
- How fast are your disks? Find out the open source way, with fio
- Performance benchmarking with Fio on Nutanix

Revision #2
Created 1 September 2024 17:02:41 by dustin@swigg.net
Updated 1 September 2024 18:07:55 by dustin@swigg.net