How to profile filesystem with fio

Practical Class 7-a

Systems and Storage Laboratory
Department of Computer Science and Engineering
Chung-Ang University

What is fio?

- **fio - Flexible I/O tester**
  - Fio spawns a number of threads or processes doing a particular type of I/O action as specified by the user
  - Fio takes a number of global parameters, each inherited by the thread unless otherwise parameters given to them overriding that setting is given
  - The typical use of fio is to write a job file (script file) matching the I/O workload that one wants to simulate

Writing fio script (job file)

- rand-write.fio

```
; -- start job file --
[global]
name=fio-rand-write
directory=/mnt/test
;filename=fio-rand-write
rw=randwrite
bs=4K
direct=0
numjobs=72
;verify=meta

[file1]
size=1G
;ioengine=libaio
group_reporting
; -- end job file --
```

Writing fio script (job file)

- **Comment**
  - If the first character in a line is a '**;**' or a '#', the entire line is discarded as a comment

```
; -- start job file --
; say hello to your fio script!
; -- end job file --
```

Writing fio script (job file)

Job name

- the names enclosed in [] brackets define the job name
- You are free to use any ASCII name you want, except *global* which has special meaning
  - A *global* section sets defaults for the jobs described in that file
  - A job may override a *global* section parameter, and a job file may even have several *global* sections if so desired
  - A job is only affected by a *global* section residing above it

```
[global]
; default parameters are to be defined here…

[job1]
; this is job1 section…

[job2]
; this is job2 section…
```

Writing fio script (job file)

❖ Job description

- name=str
  - ASCII name of the job
    - ✓ This may be used to override the name printed by fio for this job. Otherwise the job name is used.
- numjobs=int
  - Create the specified number of clones of this job
    - ✓ Each clone of job is spawned as an independent thread or process

```
[global]
name=fio-rand-write
…
numjobs=72
…
```

Writing fio script (job file)

❖ Target file/device

- directory=str
  - Prefix filenames with this directory
    - ✓ Used to place files in a different location than `./.`
- filename=str
  - Fio normally makes up a filename based on the job name, thread number, and file number
  - specify a *filename* for each of them to override the default

```
[global]
…
directory=/mnt/test
;filename=fio-rand-write
…
```

Writing fio script (job file)

- **I/O type**
  - direct=bool
    - If value is true, use non-buffered I/O
    - This is usually *O_DIRECT*
  - readwrite=str, rw=str
    - read : Sequential reads
    - write : Sequential writes
    - randread : Random reads
    - randwrite : Random writes

```
[global]
…
rw=randwrite
…
direct=0
…
```

Writing fio script (job file)

- **Block size**
  - blocksize=int[,int][,int], bs=int[,int][,int]
    - The block size in bytes used for I/O units
    - Default: 4096 (bytes)
    - A single value applies to reads, writes, and trims

```
[global]
…
bs=4K
…
```

Writing fio script (job file)

I/O engine

- ioengine=str
  - libaio
    - ✓ Linux native asynchronous I/O.
    - ✓ Note that Linux may only support queued behavior with non-buffered I/O (set `direct=1` or `buffered=0`).

```
…
[file1]
…
;ioengine=libaio
…
```

Writing fio script (job file)

❖ Verification

- verify=str
  - meta
    - ✓ meta information is included in generic verification header
    - ✓ meta verification happens by default
    - ✓ This option is kept because of compatibility's sake with old configurations

```
[global]
…
;verify=meta
```

Writing fio script (job file)

Measurements and reporting

- group_reporting
  - Displays statistics for groups of jobs as a whole instead of for each individual job
  - Especially true if *numjobs* is used

```
…
[file1]
…
group_reporting
```

Executing fio script

- **Assume rand-write.fio is a file in format of fio script**
  - First, mount the target device into desired directory
    - In this case, `/mnt/test`

```
$ sudo mount –t [mount type] /dev/[yourdevice] /mnt/test
```

  - Since we are accessing to `/mnt/test`, fio requires sudo privilege

```
$ sudo fio rand-write.fio
```

  - Check the result after fio jobs are done
  - Lastly, don’t forget to unmount your device

```
$ sudo umount /mnt/test
```

Result

- **Result screen from the rand-write.fio**

```
file1: (groupid=0, jobs=72): err= 0: pid=8047: Thu Oct 27 02:55:07 2022
  write: IOPS=355k, BW=1387MiB/s (1455MB/s)(72.0GiB/53149msec)
    clat (usec): min=4, max=631461, avg=198.43, stdev=4838.13
     lat (usec): min=4, max=631461, avg=198.55, stdev=4838.14
    clat percentiles (usec):
     |  1.00th=[    14],  5.00th=[    47], 10.00th=[    73], 20.00th=[    87],
     | 30.00th=[    96], 40.00th=[   104], 50.00th=[   112], 60.00th=[   120],
     | 70.00th=[   128], 80.00th=[   139], 90.00th=[   155], 95.00th=[   174],
     | 99.00th=[   233], 99.50th=[   281], 99.90th=[ 16712], 99.95th=[ 19006],
     | 99.99th=[325059]
   bw (  KiB/s): min=   48, max=105068, per=1.40%, avg=19886.53, stdev=8904.07, samples=7483
   iops        : min=   12, max=26267, avg=4971.34, stdev=2225.97, samples=7483
  lat (usec)   : 10=0.60%, 20=1.27%, 50=3.46%, 100=29.54%, 250=64.39%
  lat (usec)   : 500=0.52%, 750=0.06%, 1000=0.03%
  lat (msec)   : 2=0.02%, 4=0.01%, 10=0.01%, 20=0.06%, 50=0.02%
  lat (msec)   : 100=0.01%, 250=0.01%, 500=0.02%, 750=0.01%
  cpu          : usr=1.56%, sys=94.20%, ctx=649075, majf=0, minf=11443
  IO depths    : 1=100.0%, 2=0.0%, 4=0.0%, 8=0.0%, 16=0.0%, 32=0.0%, >=64=0.0%
     submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     issued rwt: total=0,18874368,0, short=0,0,0, dropped=0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1

Run status group 0 (all jobs):
  WRITE: bw=1387MiB/s (1455MB/s), 1387MiB/s-1387MiB/s (1455MB/s-1455MB/s), io=72.0GiB (77.3GB), run=53149-53149msec

Disk stats (read/write):
    md127: ios=72/16590624, merge=0/0, ticks=0/0, in_queue=0, util=0.00%, aggrios=18/4147640, aggrmerge=0/15, aggrtic
ks=1/118008459, aggrin_queue=112011997, aggrutil=45.47%
  nvme0n1: ios=0/4139953, merge=0/59, ticks=0/3992613, in_queue=809920, util=45.01%
  nvme3n1: ios=62/4151776, merge=0/3, ticks=6/226867339, in_queue=218642652, util=45.45%
```

Tips (Must Read)

- **Buffered Random Read/Write**
  - need to run twice and then use the second result
- **The numjobs**
  - must be equal to the number of CPU cores
- **Block size**
  - should be 4k
- **Run sequence read/write before random read/write.**