2024-07-30

Page Cache Benchmarking

- The original hypothesis: Page Cache is too slow to be used with modern NVMe devices.

```
$ fio --direct=0 --rw=write --size=32G --bs=1M --filename=/dev/nvme1n1p2
2952MiB/s

$ fio --direct=1 --rw=write --size=32G --bs=1M --filename=/dev/nvme1n1p2
6587MiB/s
```

Interesting params

- Parameters to control writeback when benchmarking page cache:

```
# default 10, triggers writeback when threshold reached
vm.dirty_background_ratio=80

# default 20, blocks when threshold reached and switch to direct
vm.dirty_ratio=90
```

Fio with block device

- O_DIRECT behavior is different for a block device and a regular file.
- With block device:

```
$ head -5 /proc/meminfo
MemTotal:       64951852 kB
MemFree:        29651472 kB
MemAvailable:   63573604 kB
Buffers:        33556620 kB     # !!!
Cached:           132944 kB
```

Fio with regular file

- With regular file:

```
$ head -5 /proc/meminfo
MemTotal:       64951852 kB
MemFree:        30639684 kB
MemAvailable:   63801840 kB
Buffers:            2196 kB
Cached:         33703140 kB     # !!!
```

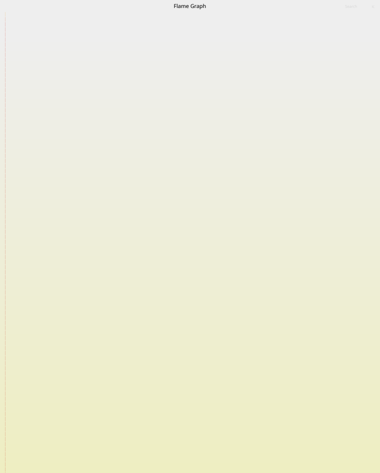
Virtual Memory

- `Cached` works as expected for a page cache.
- `Buffers` represents an IO buffer cache and does not survive longer than an issuing process.
  - This is used while updating on-disk metadata (inode tables, allocation bitmaps…).

Flamegraph: Block device

```
$ fio --direct=0 --rw=write --size=32G --bs=1M --filename=/dev/nvme1n1p2
2952MiB/s
```

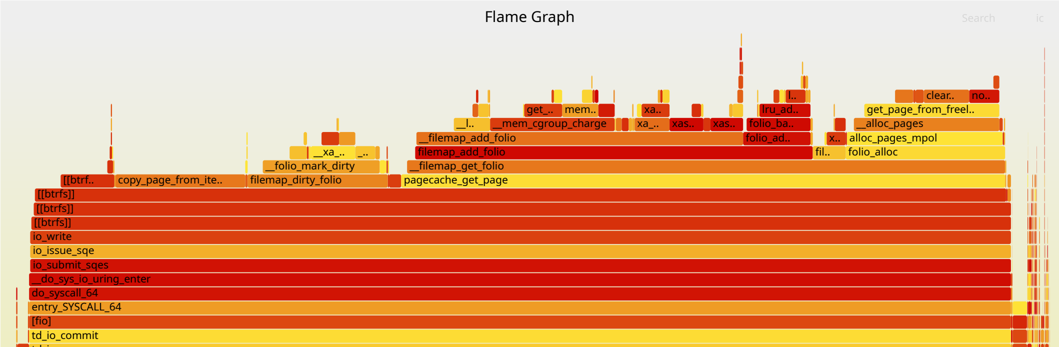
- Flamegraph link



Flamegraph: Regular File (None Cached)

```
$ echo 3 | sudo tee /proc/sys/vm/drop_caches
$ fio --direct=0 --rw=write --size=32G --bs=1M --iodepth=1 --numjobs=1 # First run (Nothing in page cache)
3795MiB/s
```

- Flamegraph link



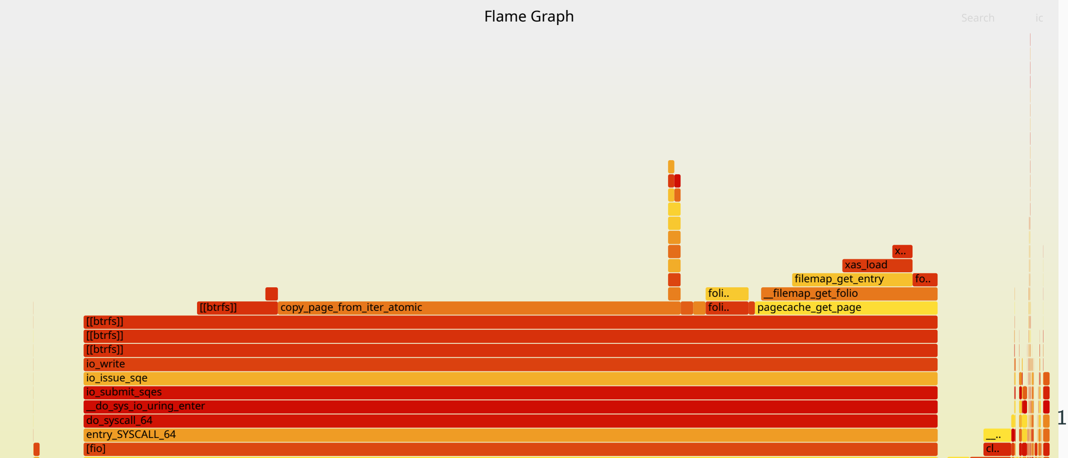
Regular File (All Cached)

```
$ fio --direct=0 --rw=write --size=32G --bs=1M --iodepth=1 --
numjobs=1 # Second write (All cached)
10.9GiB/s
$ fio --direct=0 --rw=read --size=32G --bs=1M --iodepth=1  --
numjobs=1 # Read (All cached)
18.2.GiB/s
```

- Raw memory bandwidth limit is around 96GB/s (2x DDDR5-6000 = 2x48GB/s).
- With C++ code I can read:
  - 55GB/s 1-thread
  - 70GB/s 2-threads
  - No CCD (core chiplet die) pinning

Flamegraph: Regular File (All Cached)

- Flamegraph link
- Dominated by `copy_page_from_iter_atomic` but the memory throughput is not saturated.


Flame Graph
Search
ic
x..
xas_load
filemap_get_entry
fo..
foli..
__filemap_get_folio
[[btrfs]]
copy_page_from_iter_atomic
foli..
pagecache_get_page
[[btrfs]]
[[btrfs]]
[[btrfs]]
io_write
io_issue_sqe
io_submit_sqes
__do_sys_io_uring_enter
do_syscall_64
entry_SYSCALL_64
[fio]
__..
cl..