Buffered I/O の速度の計測と考察

205723K 佐野巧曜

2023 年 1 月 8 日

実験データ

Golang の bufio パッケージで Buffered I/O を実装した。

実験の計測したときの環境は以下の通りである。

```
Operating System: Ubuntu 22.04.1 LTS
          Kernel: Linux 5.15.0-53-generic
    Architecture: x86-64
 Hardware Vendor: Dell Inc.
  Hardware Model: PowerEdge R740
     File System: ext4
      Go version: go1.19.1 linux/amd64
```

以下の 2 つのグラフがバッファなしから 16384 bytes までのバッファでファイルに書き出したときの速度のグラフである。

横軸がファイルサイズ (bytes) で縦軸が実行時間 (ナノ秒) である

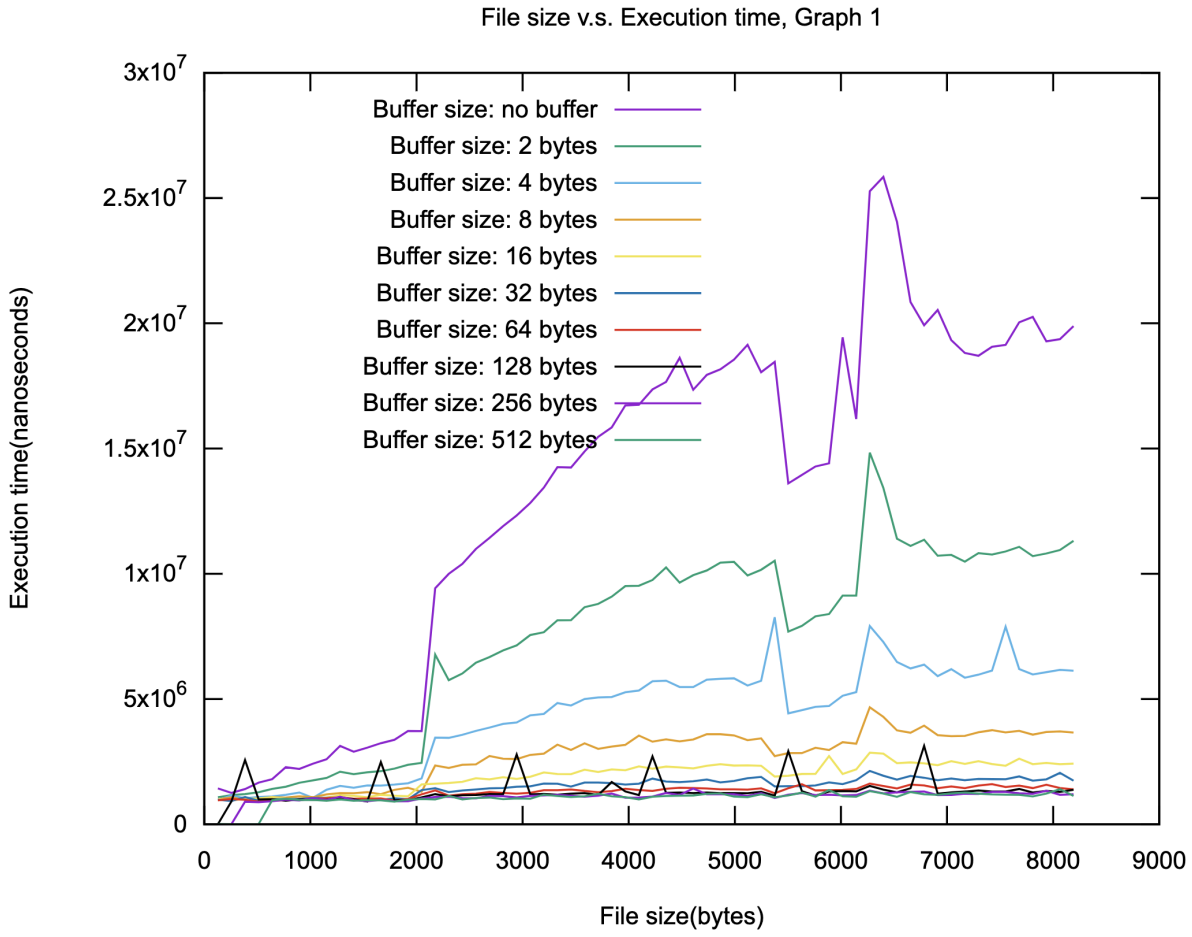

図 1: 512 bytes 以内範囲のバッファの書き出しの速度のグラフ

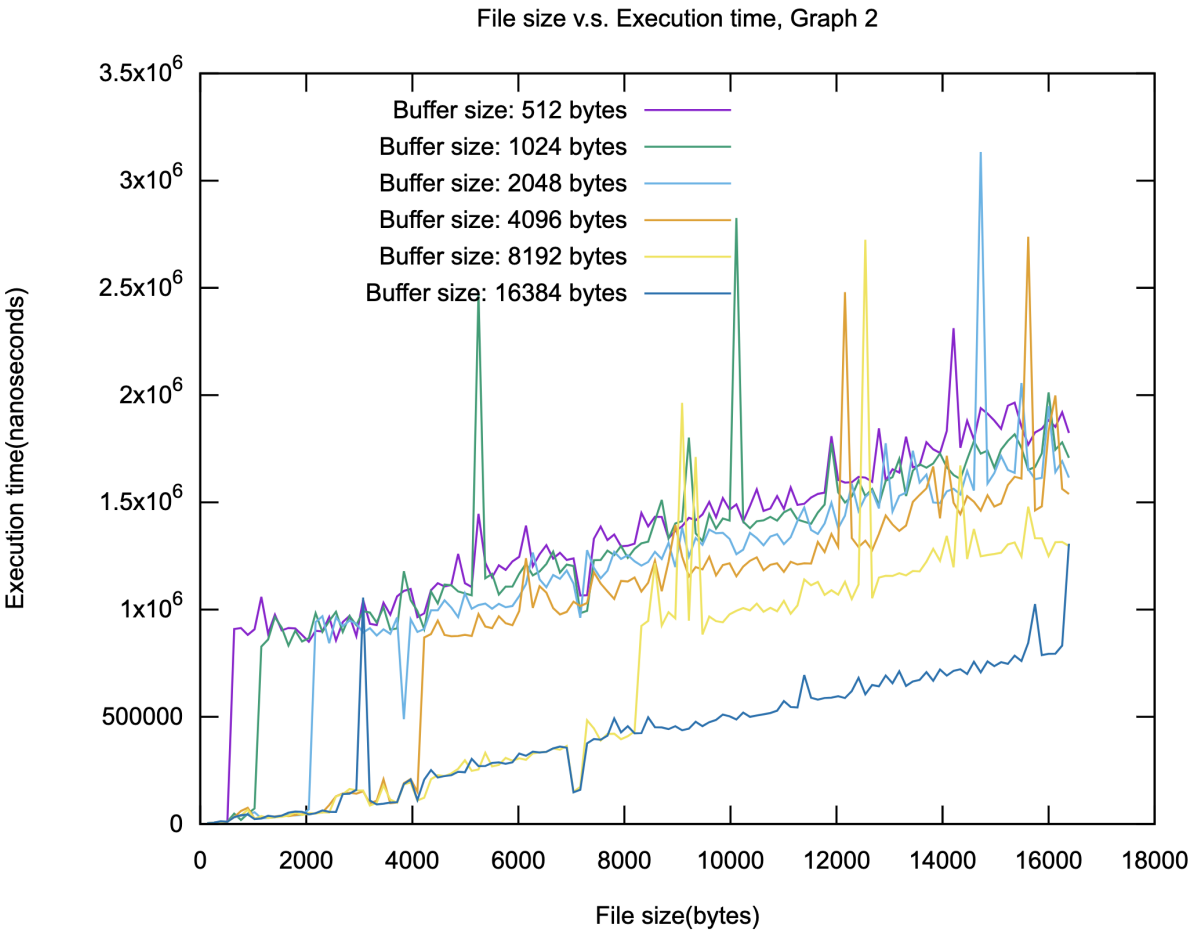

図 2: 16384 bytes 以内の範囲のバッファの書き出しの速度のグラフ

図 1 からバッファなし、バッファありの書き出しではバッファありの書き出しの方が高速であるとわかった。図 1、図 2 から、バッファのサイズがファイルサイズより大きい範囲では、バッファのサイズとファイルの書き出し速度に相関は見られなかった。また、図 2 からは、書き出すファイルのサイズがバッファサイズを超えると、書き出しに 1×10^6 付近かそれ以上の実行時間が掛かることがわかった。

考察

バッファのサイズがファイルサイズより大きい範囲でバッファのサイズとファイルの書き出し速度に相関がないことと、書き出すファイルのサイズがバッファサイズを超えるときに書き出しに 1×10^6 付近かそれ以上の実行時間が掛かることは、共通の背景があると考えられる。

bufio.Writer の Writer 関数の実装 [1] から、bufio.Writer の buf の空きがなくなるまで、buf に書き込みを行い、buf の空きがなくなったときにのみ、Flush() 関数で buf の中身をメモリに書き込むために、write(2) システムコールを呼び出すことになっている。bufio.Writer によるファイルの書き出しに、write(2) システムコールが呼ばれることは、strace コマン

ド (Linux)、dtruss コマンド (Mac OS) で検証できる。write(2) システムコールはユーザー空間のメモリ領域をカーネル空間にあるページキャシュに書き写す処理があるので、そのときにオーバーヘッドが生じる。そのためバッファリングを行って逐一 write(2) システムコールの回数を少なくすることで、書き出しの高速化ができるようになる。

また、OS のファイル API の観点からでは、Linux のファイルシステムである ext4 では、ブロックといわれる単位でファイル読み書きを扱っていて、そのブロックのサイズが 4096 bytes[2] なので、4096 bytes の自然数倍じゃないバッファサイズで書き込みをすると無駄な切り上げが生じる。なので、実験の結果とファイル API の実装の観点から、省メモリを無視した場合、ファイルの書き出しを高速化するには、バッファサイズを 4096 の自然数倍のバイトでなるべく大きく確保するべきだと考えられる。

参考文献

[1] Google.”src/bufio/bufio.go”.Google Open Source.2022-05-03. https://cs.opensource.google/go/go/+/refs/tags/go1.19.4:src/bufio/bufio.go;l=663 ,(参照 2023-01-08).
[2] Linus Torvalds.”fsverity”.GitHub.2022-06-10. https://github.com/torvalds/linux/blob/master/Documentation/filesystems/fsverity.rst#ext4 ,(参照 2023-01-08).