What posix_fadvise() args for sequential file write?

I am working on an application which does sequentially write a large file (and does not read at all), and I would like to use `posix_fadvise()` to optimize the filesystem behavior.

The function description in the manpage suggests that the most appropriate strategy would be `POSIX_FADV_SEQUENTIAL`. However, the Linux implementation description doubts that:

> Under Linux, POSIX_FADV_NORMAL sets the readahead window to the default size for the backing device; POSIX_FADV_SEQUENTIAL doubles this size, and POSIX_FADV_RANDOM disables file readahead entirely.

As I'm only writing data (overwriting files possibly too), I don't expect any readahead. Should I then stick with my `POSIX_FADV_SEQUENTIAL` or rather use `POSIX_FADV_RANDOM` to disable it?

How about other options, such as `POSIX_FADV_NOREUSE`? Or maybe do not use `posix_fadvise()` for writing at all?

c | optimization | posix

edited **Sep 20 '10 at 21:56**

asked **Sep 20 '10 at 21:44**

Michał Górny
1,101 2 13
88% accept rate

Was this post useful to you?

3 Answers

It all depends on the temporal locality of your data. If your application won't need the data soon after it was written, then you can go with `POSIX_FADV_NOREUSE` to avoid writing to the buffer cache (in a similar way as the `O_DIRECT` flag from `open()`).

answered **Sep 20 '10 at 21:53**

Trixl
36 2

feedback

Most of the `posix_fadvise()` flags (eg `POSIX_FADV_SEQUENTIAL` and `POSIX_FADV_RANDOM`) are hints about readahead rather than writing.

There's some advice from Linus here and here about getting good sequential write performance. The idea is to break the file into large-ish (8MB) windows, then loop around doing:

- Write out window N with `write()`;
- Request asynchronous write-out of window N with `sync_file_range(..., SYNC_FILE_RANGE_WRITE)`
- Wait for the write-out of window N-1 to complete with `sync_file_range(..., SYNC_FILE_RANGE_WAIT_BEFORE | SYNC_FILE_RANGE_WRITE | SYNC_FILE_RANGE_WAIT_AFTER)`
- Drop window N-1 from the pagecache with `posix_fadvise(..., POSIX_FADV_DONTNEED)`

This way you never have more than two windows worth of data in the page cache, but you still get the kernel writing out part of the pagecache to disk while you fill the next part.

answered **Sep 21 '10 at 0:13**

caf
63.7k 3 32 105

feedback

As far as writes go I think that you can just rely on the OSes disk IO scheduler to do the right thing.

You should keep in mind that while `posix_fadvise` is there specifically to give the kernel hints about future file usage patterns the kernel also has other data to help it out.

If you don't open the file for reading then it would only need to read blocks in when they were partially written. If you were to truncate the file to 0 then it doesn't even have to do that (you said that you were overwriting).

answered **Sep 20 '10 at 21:59**

nategoose
5,927 6 12

feedback

Not the answer you're looking for? Browse other questions tagged c optimization posix **or ask your own question.**

question feed