Non-blocking Async IO in Java

Asked 4 years, 8 months ago Modified 2 years, 8 months ago Viewed 2k times

1

Is there any way to do asynchronous IO in Java without blocking any threads (including background threads)? Coming from C#, my understanding of async IO is that it when you call

```
await ReadAsync()
```

The calling thread (part of a threadpool) steps into the `ReadAsync` function, at some point calls an asynchronous read function from the OS kernel, and then adds itself back to the threadpool to pick up other `Tasks`. Once the read is completed, the threadpool is notified and another thread picks up the rest of the `Task`.

In Java, on the other hand, the documentation and this answer seem to suggest that asynchronous IO functions are simply called by a background thread that then blocks. This seems less performant. Is there any way to achieve true, non-blocking IO in Java?

java asynchronous io

Share Edit Follow

edited Apr 26, 2020 at 16:00

asked Apr 26, 2020 at 15:55

A0A

118 • 6

This question needs more detail. What have you tried? What didn't work? – nicomp Apr 26, 2020 at 15:58

1 There is also WebFlux docs.spring.io/spring-framework/docs/5.0.0.BUILD-SNAPSHOT/... – Sully Apr 26, 2020 at 15:59

@HithamS.AlQadheeb thanks for the link! I'm also aware of the EA async-await library. I'm hoping to find out if there is a way to achieve this from the standard library, but will check those out if I need to. – A0A Apr 26, 2020 at 16:02

You can also check ThreadPoolExecutor docs.oracle.com/javase/7/docs/api/java/util/concurrent/... – Sully Apr 26, 2020 at 16:50

1 There is no true way for Java to provide asynchronous I/O. There is also no way for Java use Direct I/O instead of buffered I/O. The Netty project does have an iouring binding that makes both Direct I/O and asynchronous I/O possible in Java. I'm currently using the API directly in a prototype. – pveentjer May 15, 2022 at 12:30

2 Answers

Sorted by: Highest score (default)

3

The `AsynchronousFileChannel.open()` returns instances of different implementations according to the running environment. On Windows it should return an instance of `WindowsAsynchronousFileChannelImpl` which uses I/O completion port and avoids blocking threads on IO operations. Threads of thread pool are only used to dispatch results and do not block, unless the end user programmer blocks that thread.

The RxIo is built on top of AFC and provides the `AsyncFiles` equivalent to the synchronous `Files` class but with an asynchronous API. Taking advantage of the continuation-passing style of `CompletableFuture` (equivalent to .net `Task`) you may read a file content without blocking:

```
AsyncFiles
  .readAll(path)
  .thenAccept(body ->     /* invoked on completion */)
  .exceptionally(excep -> /* invoked on error*/
```

You may run the unit tests of RxIo and place a breakpoint at open() and inspect the implementation of `WindowsAsynchronousFileChannelImpl`.

Share Edit Follow

answered May 8, 2020 at 12:50

Miguel Gamboa

9,333 • 7 • 52 • 103

Got it. How is it implemented on Linux? Is it non - blocking there, too? – A0A May 9, 2020 at 14:55

2 I think not. From the observations made on this answer stackoverflow.com/questions/39501924/... it seems that on Linux it is blocking one thread per each IO operation. – Miguel Gamboa May 11, 2020 at 12:05

2

Until some time ago there were problems with asynchronous file I/O on Linux. There was an aio interface, but it was only asynchronous for O_DIRECT, which is quite inconvenient for standard use cases. So the standard JDK implementation of `AsynchronousFileChannel` for Linux internally uses thread pooling and simple blocking I/O which is not really asynchronous I/O.

Things have changed a bit since Linux introduced the `io_uring` interface. It is now possible to use real non-blocking file I/O not just for O_DIRECT but for buffered I/O too. And a lot more, to reduce overhead of syscall and increase performance. Read more about io_uring.

At the moment there is no built-in support for `io_uring` in Java. There have been rumors that support may appear for better project Loom support, but that's just a rumors.

There are third party libraries that add asynchronous file I/O support via `io_uring` for Java - jasyncfio.

Share Edit Follow

edited May 16, 2022 at 6:08

answered May 14, 2022 at 10:03

ikorennoy

230 ● 1 ● 10

Netty also get an iouring binding. Works fine with files but some tweaks are needed. Also the classes are not public so need to add the classes that use the API in the same dir (for now) – pveentjer May 15, 2022 at 4:47