



MasterT-J
码龄7年  暂无认证

251
原创

2万+
周排名

100万+
总排名

55万+
访问


等级

7168
积分

653
粉丝

445
获赞

118
评论

1593
收藏



私信

关注

搜博主文章



- 热门文章
- 计算机缓存Cache以及Cache Line详解 

66472
- 深入浅出全面解析RDMA 

63701
- 大数据采集技术综述 

26129
- 图论：连通分量和强连通分量 

19140
- Redis：一致性Hash算法 

11070

- 分类专栏
- 图计算

14篇
- Java语言系统学习

10篇
- Java高并发编程

17篇
- RDMA

RDMA

18篇
- 数据结构与算法分析

14篇
- Kafka框架

3篇
- 

- 最新评论
- Redis：一致性Hash算法

weixin_38420911: 复制粘贴别人的最好写明出处哈
- 计算机缓存Cache以及Cache Line详解

Jiakaic: 解决了吗？我也有相同的困扰
- NIO详解（六）：Java堆外内存

a desiner: 1年后咋就看不懂了呢，而且有些地方感觉。。。mapperByteBuffer
- Netty详解（一）：Reactor模型

a desiner: 讲的实在是太好了，netty权威指南误人啊，多亏看了这篇博客
- NIO详解（二）： BIO 浅谈 同步 异步与...

m0_55458338: 是一篇结合实际应用的总结，比那些列举一大堆故事例子更贴合？ ...
- 你厉害点吧七娘挂 “瑞奇”挂云” 呀？

NIO详解（十二）：AsynchronousFileChannel详解

原创

MasterT-J

2019-05-20 19:17:00

 986

 收藏 1


版权

分类专栏：

Java NIO

文章标签：

Java NIO

Java NIO 专栏收录该内容

4 订阅

13 篇文章

订阅专栏

1. 概述

Java NIO中的FileChannel是一个连接到文件的通道。可以通过文件通道读写文件。FileChannel无法设置为非阻塞模式，他总是运行在阻塞模式下。在Java 7中，AsynchronousFileChannel被添加到Java NIO。AsynchronousFileChannel使读取数据，并异步地将数据写入文件成为可能。

2. 创建一个AsynchronousFileChannel

使用AsynchronousFileChannel提供的静态方法 open() 创建它。示例代码如下：

```
1 Path path = Paths.get("data/test.xml");
2 AsynchronousFileChannel fileChannel =AsynchronousFileChannel.open(path, StandardOpenOption.READ);
```

第一个参数是一个 PATH 的对象实例，它指向了那个与 AsynchronousFileChannel 相关联的文件。

第二个参数是一个或多个操作选项，它决定了 AsynchronousFileChannel 将对目标文件做何种操作。示例代码中我们使用了 StandardOpenOption.READ，它表明我们将要对目标文件进行读操作。

3. 从AsynchronousFileChannel中读取数据

3.1 使用Futtrue读取数据

从AsynchronousFileChannel读取数据的第一种方法是调用返回Future的read()方法。下面是如何调用这个read()方法的示例:

```
1 Future<Integer> operation = fileChannel.read(buffer, 0);
```

read()方法的这个版本将ByteBuffer作为第一个参数。从AsynchronousFileChannel读取的数据被读入这个ByteBuffer。第二个参数是文件中的字节位置，以便开始读取。

read()方法会立即返回，即使读操作还没有完成。通过调用read()方法返回的Future实例的isDone()方法，您可以检查读取操作是否完成。

```
1 AsynchronousFileChannel fileChannel =
2     AsynchronousFileChannel.open(path, StandardOpenOption.READ);
3
4 ByteBuffer buffer = ByteBuffer.allocate(1024);
5 long position = 0;
6
7 Future<Integer> operation = fileChannel.read(buffer, position);
8
9 while(!operation.isDone());
10
11 buffer.flip();
12 byte[] data = new byte[buffer.limit()];
13 buffer.get(data);
14 System.out.println(new String(data));
15 buffer.clear();
```

MasterT-J

关注

 0


 3


 1






专栏目录

疯狂盲盒





举报

强烈不推荐

不推荐

一般般

推荐

强烈推荐

最新文章

一周一论文（翻译）——[SIGMOD 2015]

TIMELY RTT-based Congestion Control for the Datacenter

一周一论文（翻译）——[SIGMOD 2015]

Congestion Control for Large-Scale RDMA

一周一论文（翻译）——[SIGMOD 2016]

RDMA over Commodity Ethernet at Scale

2020年 8篇

2019年 112篇

2018年 136篇

目录

1. 概述

2. 创建一个AsynchronousFileChannel

3. 从AsynchronousFileChannel中读取数据

3.1 使用Futtrue读取数据

3.2 使用CompletionHandler读取数据

4. 向AsynchronousFileChannel中写入数据

4.1 使用Future读取数据

4.2 使用CompletionHandler写入数据

上面的程序首先创建了一个 AsynchronousFileChannel 对象，然后调用它的read()方法返回一个Future。其中 read()方法需要两个参数，一个是ByteBuffer，另一个是读取文件的开始位置。然后通过循环调用isDone() 方法检测读取过程是否完成，完成后 isDone()方法将返回true。尽管这样让cpu空转了一会，但是我们还是应该等读取操作完成后再进行后续的步骤。

一旦读取完成，数据被存储到ByteBuffer，然后将数据转化为字符串既而输出。

3.2 使用CompletionHandler读取数据

第二种读取数据的方式是调用AsynchronousFileChannel 的另一个重载 read() 方法，改方法需要一个 CompletionHandler 作为参数。下面是代码示例：

```

1  fileChannel.read(buffer, position, buffer, new CompletionHandler<Integer, ByteBuffer>() {
2      @Override
3      public void completed(Integer result, ByteBuffer attachment) {
4          System.out.println("result = " + result);
5
6          attachment.flip();
7          byte[] data = new byte[attachment.limit()];
8          attachment.get(data);
9          System.out.println(new String(data));
10         attachment.clear();
11     }
12
13     @Override
14     public void failed(Throwable exc, ByteBuffer attachment) {
15
16     }
17 });

```

一旦读取操作完成，CompletionHandler的 complete() 方法将会被调用。它的第一个参数是个 Integer类型，表示读取的字节数。第二个参数 attachment 是 ByteBuffer 类型的，用来存储读取的数据。它其实就是由 read() 方法的第三个参数。当前示例中，我们选用 ByteBuffer 来存储数据，其实我们也可以选用其他的类型。读取失败的时候，CompletionHandler的 failed()方法会被调用。

4. 向AsynchronousFileChannel中写入数据

就像读取一样，我们同样有两种方式向 AsynchronousFileChannel 写入数据。我们可以调用它的2个重载的 write() 方法。下面我们将分别加以介绍。

4.1 使用Future读取数据

AsynchronousFileChannel也可以异步写入数据。下面是一个完整的写入示例：

```

1  Path path = Paths.get("data/test-write.txt");
2  AsynchronousFileChannel fileChannel =
3      AsynchronousFileChannel.open(path, StandardOpenOption.WRITE);
4
5  ByteBuffer buffer = ByteBuffer.allocate(1024);
6  long position = 0;
7
8  buffer.put("test data".getBytes());
9  buffer.flip();
10
11 Future<Integer> operation = fileChannel.write(buffer, position);
12 buffer.clear();
13
14 while(!operation.isDone());
15
16 System.out.println("Write done");

```

首先实例化一个写入模式的 AsynchronousFileChannel, 然后创建一个 ByteBuffer 并写入一些数据。再然后将数据写入文件。最后，检查返回的 Future，看是否写入完成。

注意，写入目标文件要提前创建好，如果它不存在的话，`writhe()` 方法会抛出一个 `java.nio.file.NoSuchFileException`。

4.2 使用CompletionHandler写入数据

使用 CompletionHandler代替Future向AsynchronousFileChannel写入数据，这种方式可以更加直接的知道写入过程是否完成。下面是示例程序：

```

1 Path path = Paths.get("data/test-write.txt");
2 if(!Files.exists(path)){
3     Files.createFile(path);
4 }
5 AsynchronousFileChannel fileChannel =
6     AsynchronousFileChannel.open(path, StandardOpenOption.WRITE);
7
8 ByteBuffer buffer = ByteBuffer.allocate(1024);
9 long position = 0;
10
11 buffer.put("test data".getBytes());
12 buffer.flip();
13
14 fileChannel.write(buffer, position, buffer, new CompletionHandler<Integer, ByteBuffer>() {
15
16     @Override
17     public void completed(Integer result, ByteBuffer attachment) {
18         System.out.println("bytes written: " + result);
19     }
20
21     @Override
22     public void failed(Throwable exc, ByteBuffer attachment) {
23         System.out.println("Write failed");
24         exc.printStackTrace();
25     }
26 });

```

当写入程序完成时，CompletionHandler的completed()方法将会被调用，相反如果写入失败则会调用failed()方法。

Django-Vue-Extend:基于 django2.x + vue2.x 的集成项目-源码 05-04
django-vue-extend 基于 django 2.x + vue2.x 的集成项目 技术栈: 基于 Django 2.X + django-rest-framework 前...

Java NIO 学习笔记 (六) ---异步文件通道 `AsynchronousFileChannel` czwbig 的博客 324

目录: [Java NIO 学习笔记 \(一\) ---概述, Channel/Buffer](#) [Java NIO 学习笔记 \(二\) ---聚集和分散, 通道到通道](#) [Java NI...](#)

优质评论可以帮助作者获得更高权重

H _Cade_: CompletionHandler只是读取一次，那后面的咋办 3月前 回复 ...

H _Cade_ 回复: 我是在里面继续调用read。什么场景下会用到 3月前 回复 ...

 吃一口瓜: 你是搬运工吗 1 年前 回复 ... 

Java NIO系列七之 `AsynchronousFileChannel` 异步文件通... 8-2

Java7中新增了`AsynchronousFileChannel`作为`nio`的一部分。`AsynchronousFileChannel`使得数据可以进行异步读写。下面...

...AsynchronousFileChannel_weixin_39839968的博客 8-7

在Java 7 中,AsynchronousFileChannel 已添加到 Java NIO 中,它可以异步读取数据并将数据写入文件。先说明,异步和阻塞/...

六种边缘检测算子 09-06

用六种算子(分别是gabor、拉普拉斯、priwitt、robert、sobel、wallis),对三种图象进行边缘检测比较,强烈推荐哦,是本人...

01-04 专栏目录 的图片资源以及数据库文件,大三上学期...

MasterT-J [关注](#)

 0
  3
  1
 
