链滴

多线程拷贝文件比较慢？ 求帮忙分析

作者：pangwen
原文链接：https://ld246.com/article/1493032919014
来源网站：链滴

```
package com.pangwen.usefultools.io;

import java.io.*;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;

/**
* 文件拷贝工具类 * Created on 2017/4/18.
*
* @author pangwen
* @version 0.1
*/public final class FileCopyUtils {

  /**
* 大于500M的文件为大文件 */  private static final long BIG_FILE_SIZE = 1024 * 1024 * 500;
/**
* byte[]默认长度为1024 */  private static final int BUFFER_SIZE = 1024;
/**
* 最大线程数 */  private static final int MAX_THREAD_NUM = 5;

/**
* 静态内部类实现多线程 */  private static class FileCopyWorker implements Runnable {
      private final File srcFile;
private final File targetFile;
private final long startPosition;
private final long endPosition;

/**
* constructor * * @param srcFile 源文件
* @param targetFile 目标文件
* @param startPosition 文件开始位置
* @param endPosition 文件结束位置
*/  public FileCopyWorker(final File srcFile, final File targetFile, final long startPosition, final lo
g endPosition) {
          this.srcFile = srcFile;
this.targetFile = targetFile;
this.startPosition = startPosition;
this.endPosition = endPosition;
}

      //@Override
public void run() {

          RandomAccessFile rin = null;
RandomAccessFile rout = null;
try {
              rin = new RandomAccessFile(srcFile, "r");
rin.seek(startPosition);
rout = new RandomAccessFile(targetFile, "rw");
rout.seek(startPosition);
byte[] buffer = new byte[BUFFER_SIZE];
int i;
int readLength = 0;
```

```
while ((i = rin.read(buffer)) != -1 && startPosition + readLength <= endPosition) {
                rout.write(buffer, 0, i);
readLength += i;
}
        } catch (IOException e) {
            e.printStackTrace();
} finally {
            try {
                if (null != rin)
                    rin.close();
} catch (Exception e) {
                e.printStackTrace();
}
            try {
                if (null != rout)
                    rout.close();
} catch (Exception e) {
                e.printStackTrace();
}
        }

    }
  }

  /**
* nio拷贝文件 * * @param srcFile 源文件
* @param targetFile 目标文件
*/
    public static void copyFileNio(final File srcFile, final File targetFile) throws FileNotFoundExc
ption {
      if (null == srcFile)
          throw new FileNotFoundException("src file not found!");
makeParentDirs(targetFile);
FileInputStream in = new FileInputStream(srcFile);
FileOutputStream out = new FileOutputStream(targetFile);
//获取通道
FileChannel inChannel = null;
FileChannel outChannel = null;
try {
        inChannel = in.getChannel();
outChannel = out.getChannel();
//创建缓冲区
ByteBuffer buffer = ByteBuffer.allocate(1024);
//将数据读入缓冲区
while (inChannel.read(buffer) != -1) {
            //flip() 方法让缓冲区可以将新读入的数据写入另一个通道。
buffer.flip();
//将缓冲区数据写入文件
outChannel.write(buffer);
//clear() 方法重设缓冲区，使它可以接受读入的数据。
buffer.clear();
}
      } catch (IOException e) {
          e.printStackTrace();
```

```
} finally {
        if (inChannel != null) {
            try {
                in.close();
} catch (IOException e) {
                e.printStackTrace();
}
        }
        if (outChannel != null) {
            try {
                outChannel.close();
} catch (IOException e) {
                e.printStackTrace();
}
        }
    }
  }

  /**
* 拷贝文件 * * @param srcFile 源文件
* @param targetFile 目标文件
* @param allowMultipleThread 是否开启多线程
* @throws FileNotFoundException
*/  public static void copyFile(final File srcFile, final File targetFile, final boolean allowMultiple
hread) throws FileNotFoundException {

    if (null == srcFile)
        throw new FileNotFoundException("src file not found!");
//创建父文件夹
makeParentDirs(targetFile);
long srcFileLength = srcFile.length();
if (allowMultipleThread && srcFileLength > BIG_FILE_SIZE) {
        try {
            //大文件调用多线程
copyFileMultipleThread(srcFile, targetFile);
return;  } catch (Exception e) {
            e.printStackTrace();
//多线程拷贝文件失败时调用单线程拷贝文件
copyFile(srcFile, targetFile, false);
}
    }
    FileInputStream in = null;
FileOutputStream out = null;
try {
        in = new FileInputStream(srcFile);
out = new FileOutputStream(targetFile);
byte[] buffer = new byte[BUFFER_SIZE];
int i;
while ((i = in.read(buffer)) != -1) {
            out.write(buffer, 0, i);
}
    } catch (IOException e) {
        e.printStackTrace();
} finally {
```

```
        try {
            if (null != in)
                in.close();
} catch (Exception e) {
            e.printStackTrace();
}
        try {
            if (null != out)
                out.close();
} catch (Exception e) {
            e.printStackTrace();
}
    }
  }

  /**
* 多线程拷贝文件 RandomAccessFile * * @param srcFile 源文件
* @param targetFile 目标文件
*/  private static void copyFileMultipleThread(final File srcFile, final File targetFile) {

     final long srcFileLength = srcFile.length();
int threadNum = (int) (srcFileLength / BIG_FILE_SIZE);
if (threadNum > MAX_THREAD_NUM)
        threadNum = MAX_THREAD_NUM;
long residuumFileLength = srcFileLength % threadNum;
//每份文件大小
long perFileSize = (srcFileLength - residuumFileLength) / threadNum;
//开始位置
long startPosition = 0;
//结束位置
long endPosition = perFileSize;
for (int i = 0; i < threadNum; i++) {
        new Thread(new FileCopyWorker(srcFile, targetFile, startPosition, endPosition)).start();
//下一现场读取文件开始位置
startPosition = endPosition + 1;
//下一现场读取文件结束位置
endPosition += perFileSize;
//最后一个线程读取到文件末
if (i == threadNum - 2)
            endPosition = srcFileLength;
}
  }

  private static void makeParentDirs(final File file) throws FileNotFoundException {
     if (null == file)
        throw new FileNotFoundException("target file must not be null!");
File parent = file.getParentFile();
if (!parent.exists())
        parent.mkdirs();
}

  private FileCopyUtils() {
     throw new IllegalAccessError("can not create instance!");
}
```

```
}
```

原文链接：多线程拷贝文件比较慢？ 求帮忙分析