文件读写性能分析

在本次文件读写性能分析中，我们通过使用 Java 中的 DataOutputStream 和 OutputStreamWriter 分别进行大量数据写入，并测试它们在读取操作上的性能差异。实验的目标是写入和读取 5 百万个 （5 千万时间有点长）double 类型的数据，并比较两种流的执行时间。

实验过程

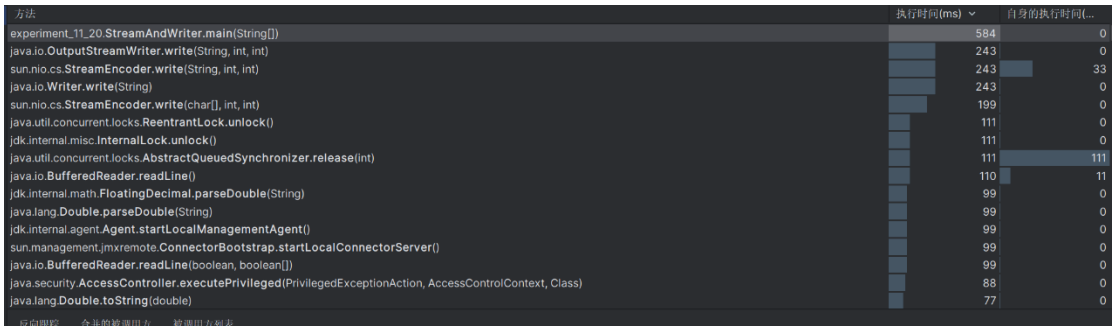
首先，我们使用 DataOutputStream 将大量 double 数据写入一个文件。接着，我们使用 OutputStreamWriter 将相同的数据写入另一个文件。接下来，我们分别使用 DataInputStream 和 InputStreamReader 读取先前写入的文件中的数据。

实验结果

通过对比使用 DataOutputStream 和 OutputStreamWriter 进行数据写入和读取的时间性能，通过 print 运行时间发现，在大量数据的情况下，DataOutputStream 明显优于 OutputStreamWriter。

```
Time taken to write to dataOutputStreamFile.txt = 16600 milliseconds
Time taken to write to outputStreamWriterFile.txt = 779 milliseconds
Time taken to read from dataOutputStreamFile.txt = 8414 milliseconds
Time taken to read from outputStreamWriterFile.txt = 241 milliseconds
```

并通过 Profiler 打印出 cpu 时间



方法	执行时间(ms)	自身的执行时间(...
experiment_11_20.**StreamAndWriter.main**(String[])	584	0
java.io.**OutputStreamWriter.write**(String, int, int)	243	0
sun.nio.cs.**StreamEncoder.write**(String, int, int)	243	33
java.io.**Writer.write**(String)	243	0
sun.nio.cs.**StreamEncoder.write**(char[], int, int)	199	0
java.util.concurrent.locks.**ReentrantLock.unlock**()	111	0
jdk.internal.misc.**InternalLock.unlock**()	111	0
java.util.concurrent.locks.**AbstractQueuedSynchronizer.release**(int)	111	111
java.io.**BufferedReader.readLine**()	110	11
jdk.internal.math.**FloatingDecimal.parseDouble**(String)	99	0
java.lang.**Double.parseDouble**(String)	99	0
jdk.internal.agent.**Agent.startLocalManagementAgent**()	99	0
sun.management.jmxremote.**ConnectorBootstrap.startLocalConnectorServer**()	99	0
java.io.**BufferedReader.readLine**(boolean, boolean[])	99	0
java.security.**AccessController.executePrivileged**(PrivilegedExceptionAction, AccessControlContext, Class)	88	0
java.lang.**Double.toString**(double)	77	0

原因如下：

二进制数据写入效率高：DataOutputStream 直接以二进制形式写入原始数据，无需进行额外的字符编码和转换。

字符编码开销：OutputStreamWriter 专注于字符流，因此在写入基本数据类型时需要将其转换为字符，可能导致性能下降。