**并行程序设计大作业报告**

范洪宇 软博18班 2018312541

1. **选题介绍**

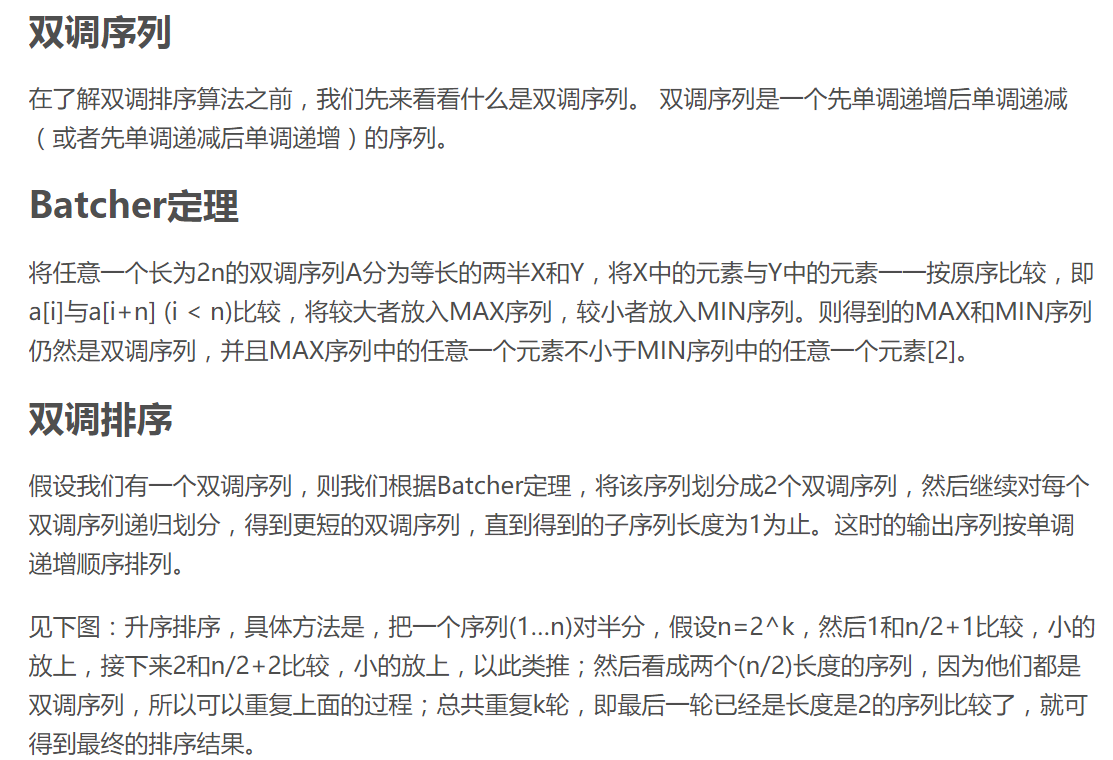
* **分段双调排序算法**

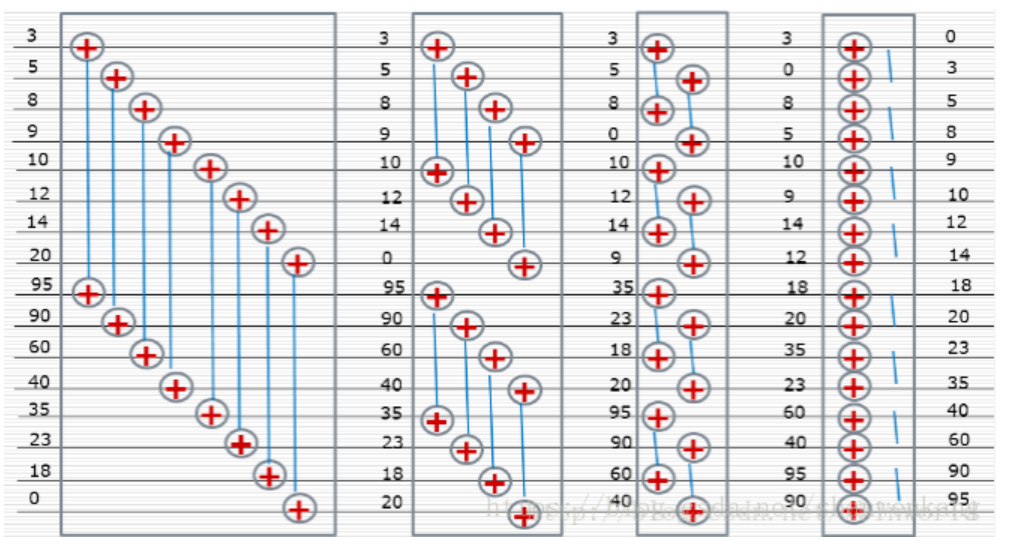
分段双调排序算法是一种非常适合并行化的排序算法，算法即给出分成m段的n个浮点数，输入的数据按照段号有序，但是每个段内部是无序的。经过分段双调排序算法处理后，每个段中数据按照递增或者递减的顺序排序。在单线程情况下，算法的时间复杂度是，并行处理后可以大大加速。

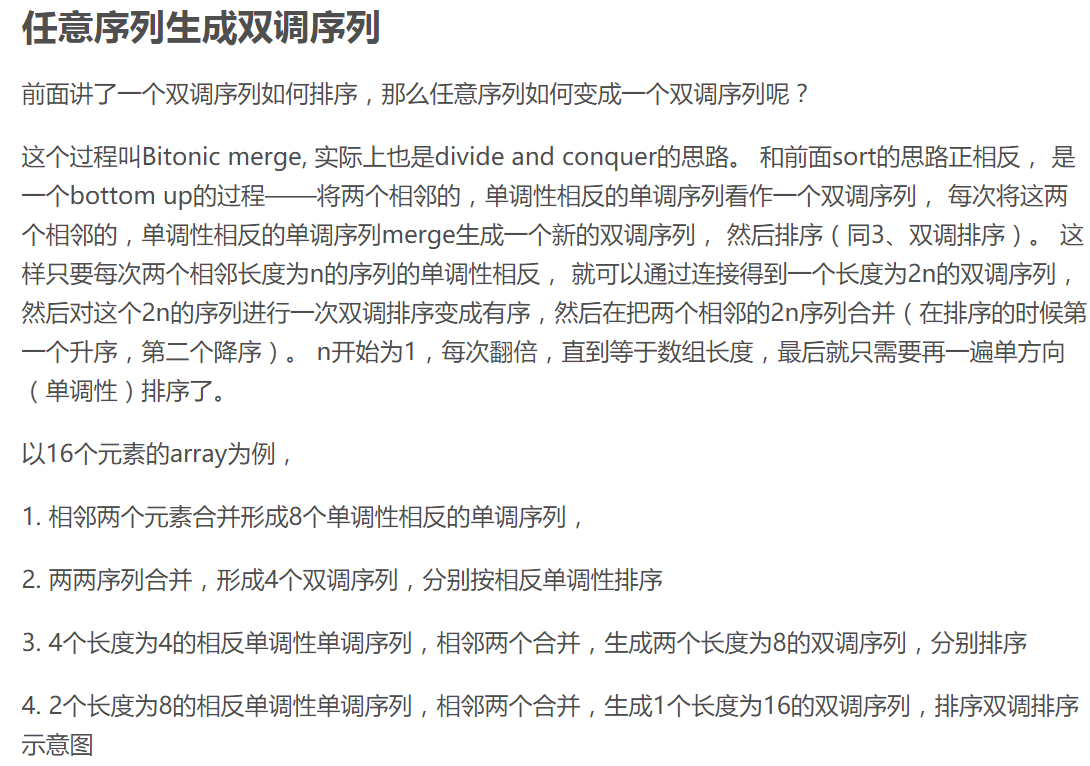
因为分段双调排序段间不影响的特点，非常适合并行化，常被并行化后用在网络拓扑排序中。除了并行化，大作业中还有一些创新点，除了输入数据申请空间进行存储，在排序的过程中算法并未申请任何额外的内存空间，仅借助已有的内存空间进行排序，从而能够保证空间安全。

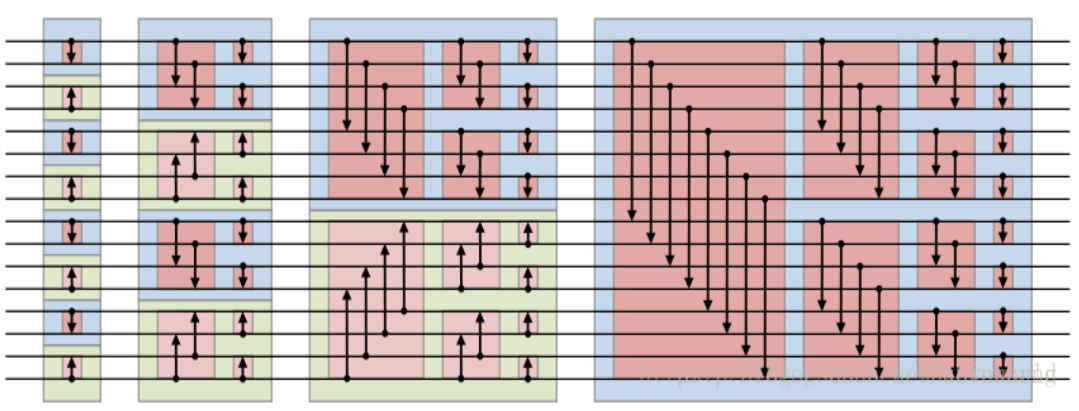
* **双调排序算法**

分段双调排序每个段之间没有数据关联，排序算法执行过程中，每个段都是使用的双调排序算法，下面会对双调排序算法进行一个简单的介绍，资料来源(https://blog.csdn.net/shanwenkang/article/details/82811130)：

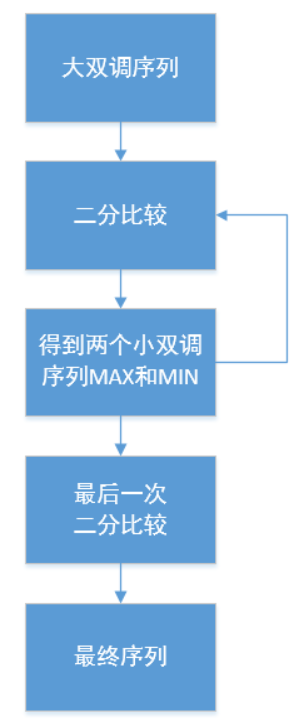
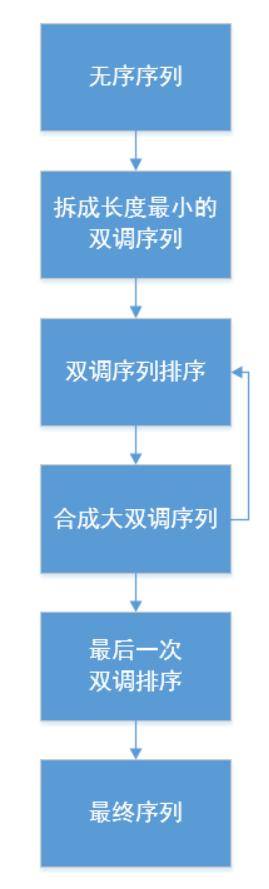








算法实现的核心流程如下左侧的流程图所示，双调序列排序每一次处理双调序列的过程如下图右图所示：



* **并行化选择**

本作业选择使用C++语言实现，使用的多线程库函数为Pthread库，为了最后代码检查简单，所有的代码实现在一个文件中，支持g++的操作系统都可以编译运行。

1. **算法实现环境以及实现方案**

* **运行平台及环境**

实验测试的台式机环境是Ubuntu 16.04 LTS，CPU配置为Inter-i7-7700，16GB内存，240GB SSD。

实验测试使用的服务器环境是Ubuntu 16.04 LTS，CPU配置为Inter至强，32GB内存，2TB HD。

实验基本上是在命令行里面运行的，调试环境使用到了Vscode。实验中使用g++编译即可，无需其他依赖。此外因为台式机和服务器上面配置不同，不同机器上实验数据不具备可比性，我们只把同一台机器上获取的结果进行对比分析。

* **实现方案**

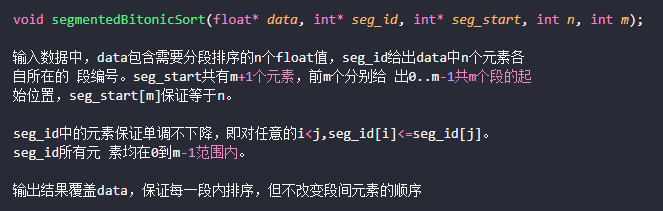
算法的原理和简单的例子已经在第一章介绍双调排序中给出了，这里不再重复介绍，仅介绍一些实现的技术细节和流程。

体现并行排序算法的提升需要一定规模的数据，所以首先是数据生成问题。

算法中实现了生成数据的部分，生成的数据作为排序算法的输入。生成数据可以指定生成数据的个数以及分段的段数，比如设置生成1千万个数据，分成10个段。生成的数据会保存在data.txt文件中，其中包括1千万个数据以及每个数据所在的段的信息。

接下来data.txt会作为排序算法的输入，因为体现并行的优势，所以进行了单线程和多线程情况的对比，最后会生成三个输出文件，分别为output\_single.txt，output\_mt.txt以及result.txt。output\_single.txt文件是在单线程情况下排序后的输出文件，output\_mt.txt文件中是在多线程线程下排序后的输出文件，这两个文件的内容是完全一样的。result.txt中存放了两种不同情况下运行时间以及耗时对比。

程序的核心接口描述如下：



程序的核心并行单元为Void\* bitonicSort(Void\* para)，其中封装了双调排序的算法实现，给定无序数组段的起始和结束位置，算法运行后能够给出排序后的数组。大作业的创新点体现在，排序算法会利用输入数据的数组空间进行排序，在整个排序的过程中不会动态申请任何的内存空间，防止不安全的访问，能够在并行网络拓扑环境中限制空间申请。

算法中同时进行了单线程排序以及多线程排序，线程的数量可以提前设置，如果线程设置数量多于段的数量，则线程数量即为段的数量，在运行过程中保证线程数量不会超过CPU支持的线程数量。

算法中还提供了格式化输出和时间监控的部分，能够将单线程和多线程情况下的排序结果输出到文件，并且对不同过程的时间进行记录和对比。

1. **实验结果对比分析**

* **1千万数据20数据段 && 单线程VS双线程**

按照该Jpg图像处理库的使用方法，我找了若干输入文件，并且借助Fuzz工具进行了评测。因为该库函数支持很多的功能，我测试了其中的向图片中插入文字信息，以及将图片中文字信息读取进行展示两个简单的功能，最后实验运行的结果如图1所示:

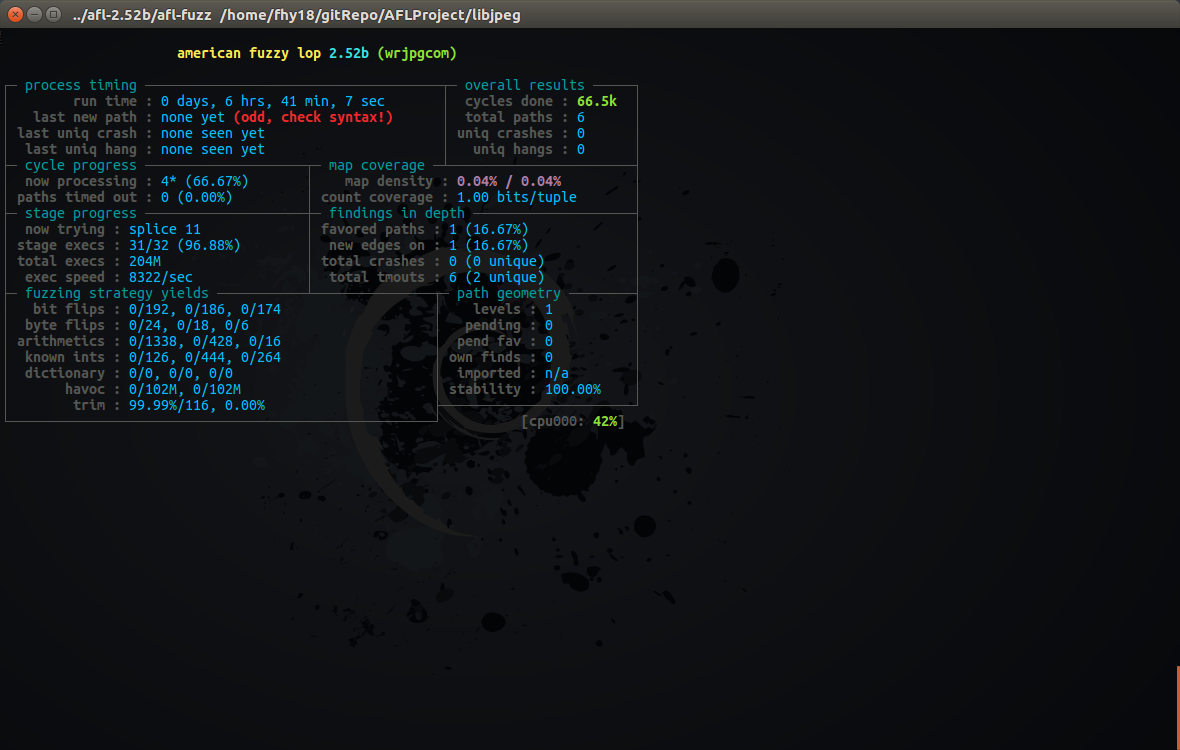
****

图 1 Libjpg项目评测结果

Fuzz程序运行了6小时40分钟，最后并没有找到crash，所以在目前的评测结果下，可以说该库函数在程序的正确性和鲁棒性方面是比较可靠的。项目运行过程中各个参数的直观情况如图2所示：

对于没有找到crash的原因可以进行简单的分析，首先是测试功能为向给定的图片中插入文字信息，插入的格式可以是任意的字符串格式的输入，因为是库函数，被很多人使用，程序本身对于文字信息类型等做了很多的预期检查，将错误情况都捕捉和排除了，所以借助AFl工具进行变种和评测并没有生成能够使该项目崩溃的变种输入。

* **1千万数据20数据段 && 单线程VS十线程**

该项目自带函数调用的入口，所以只需要设计测试用例输入即可。第一个参数表示矩阵的阶数，第二个参数表示设置的线程数目，下面紧跟的是需要计算的两个矩阵的数值信息。调用AFL工具进行Fuzz过程的实验结果如下图3所示，可以预见的是因为输入格式要求比较多，所以在Fuzz进行输入变种的过程中肯定会造成程序的崩溃：

* **1亿数据20数据段 && 单线程VS十线程**

对于没有找到crash的原因可以进行简单的分析，首先是测试功能为向给定的图片中插入文字信息，插入的格式可以是任意的字符串格式的输入，因为是库函数，被很多人使用，程序本身对于文字信息类型等做了很多的预期检查，将错误情况都捕捉和排除了，所以借助AFl工具进行变种和评测并没有生成能够使该项目崩溃的变种输入。

* **1亿数据20数据段 && 单线程VS二十线程**

对于没有找到crash的原因可以进行简单的分析，首先是测试功能为向给定的图片中插入文字信息，插入的格式可以是任意的字符串格式的输入，因为是库函数，被很多人使用，程序本身对于文字信息类型等做了很多的预期检查，将错误情况都捕捉和排除了，所以借助AFl工具进行变种和评测并没有生成能够使该项目崩溃的变种输入。

1. **总结和感想**

经过对分段双调排序的实现以及并行化处理，还是学到了并行程序设计实现方面的一些知识和方法，并且自己动手收获很多。

首先是双调排序的时间复杂度为，算不上是性能最优的排序算法，但是其先天的可并行化处理的优势使其经常被使用在并行环境之下，综合性能良好。其次的感受就是使用多个线程，受限于线程调度，资源竞争，数据传输等影响，提升的效率是要小于预期的，这也是可以预见的。

大作业实现过程中还是遇到了很多的问题，最难的在于算法的实现，其次就是基于创新点，不动态申请任何内存空间，需要对现有的数据空间进行划分，在排序过程中不断地覆盖和重复利用，保证里空间访问的安全性。在这一点的实现上调试花费了比较多的时间。

最后感谢向东老师的授课，以及助教的辛勤付出，通过《并行程序设计》这门课，复习了很多之前学习过的算法，排序，矩阵运算，图论中的一些方法等，而且对于并行程序实现进行了一些了解和动手实践，收获很多。