基于蛇型磁带的海量数据排序算法*

李建中[1,2], 张艳秋[1+]

[1](哈尔滨工业大学 计算机科学与技术学院,黑龙江 哈尔滨 150001)

[2](黑龙江大学 计算机科学技术学院,黑龙江 哈尔滨 150080)

A Massive Data Sort Algorithm Based on Serpentine Tape

LI Jian-Zhong[1,2], ZHANG Yan-Qiu[1+]

[1](School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

[2](School of Computer Science and Technology, Heilongjiang University, Harbin 150080, China)

+ Corresponding author: Phn: 86-451-6415827, Fax 86-451-6415827, E-mail: sarai@263.net

http://www.hit.edu.cn

Received 2002-03-18; Accepted 2002-08-14

Li JZ, Zhang YQ. A massive data sort algorithm based on serpentine tape. *Journal of Software*, 2003,14(1): 28~34.

Abstract: In order to solve massive data sort in digital library and dataware house, a new highly efficient algorithm based on serpentine tape, named STESort (serpentine tape external sort), is provided in this paper. Taking full advantage of the characteristics of serpentine tape, the STESort algorithm reduces the whole seek time on tapes compared with traditional 2-way merge tape sort algorithm. Besides increasing the efficiency of tape sort, the STESort algorithm prolongs the duration of tapes by reducing the times of tape header moving on tape surface. The theoretical analysis and the experimental results show that the STESort algorithm is more efficient than the traditional tape sort algorithms. The STESort is suitable for massive data sort.

Key words: serpentine tape; massive data; tape sort algorithm; STESort (serpentine tape external sort)

摘 要: 在数字图书馆和数据仓库中,需要解决海量数据的排序问题.利用蛇型磁带自身的物理特点,实现了一种高效的磁带排序算法 STESort(serpentine tape external sort).与传统的 2 路归并磁带排序算法相比,STESort 算法减少了磁带总定位时间.STESort 算法具有更优的效率.STESort 算法在提高排序效率的同时,通过减少磁头在磁带表面的移动次数延长了磁带的使用寿命.理论分析和实验结果表明,STESort 算法优于传统的磁带排序算法,适合于海量数据排序.

关键词: 蛇型磁带;海量信息;磁带排序算法;STESort(serpentine tape external sort)

中图法分类号: TP301 **文献标识码**: A

* Supported by the National Natural Science Foundation of China under Grant No.60273082 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2001AA41541 (国家高技术研究发展计划); the National Grand Fundamental Research 973 Program of China under Grant No.G1999032704 (国家重点基础研究发展规划(973))

第一作者简介: 李建中(1950－),男,黑龙江哈尔滨人,教授,博士生导师,主要研究领域为数据库系统技术,并行计算技术.

随着网络技术和信息数字化进程的推进,高于 10^{12} 字节数量级的海量数据库已经成为实际应用中常见的数据库.虽然磁盘存储系统越来越便宜,但是存储海量数据的成本仍然令人难以承受.因此,更多应用领域开始选择机器手臂磁带库保存海量信息.磁带数据的高效组织和管理向我们提出了新的挑战.磁带海量数据排序就是其中之一.排序是计算机系统中最常用的算法.20 世纪 60 年代末和 70 年代初,Knuth 在《程序设计的艺术》第 3 卷中总结了基于多个磁带驱动器的磁带数据排序算法,并对各种算法进行了详细的描述和性能上的对比[1].目前很多研究者开始研究海量数据排序算法,但主要集中在磁盘数据排序算法的研究上.文献[2]提出了单处理机环境下海量磁盘数据的排序算法.文献[3~7]提出了一系列并行磁盘数据排序算法.文献[7]中涉及到了一些磁带外排序问题,但只是简单地将传统 2 路磁带归并排序算法在多处理机上进行了并行化.

随着磁带技术的发展,新型磁带的物理特点和存取特点与传统磁带相比有很大的不同.由于传统磁带容量小、成本低,一个计算机系统通常配置多个磁带驱动器,适合于 Knuth 提出的基于多驱动器的磁带数据排序算法.现代磁带技术成本较高,从成本上考虑,我们希望排序算法能够使用较少的磁带驱动器完成排序操作.本文利用蛇型磁带的特性,用两个蛇型磁带驱动器实现了 *K*-路归并排序(*K* 通常大于 2).该排序算法使用很少的定位时间,在提高磁带排序算法效率的同时,减少磁带磨损,延长了磁带使用寿命.本文提出的磁带外排序算法称作蛇型磁带外排序算法 STESort(serpentine tape external sort).STESort 算法对于数字图书馆、数据仓库、Web 数据管理等大规模海量数据系统中的海量数据管理与查询都具有实际应用价值,为海量数据索引技术的研究提供了基础.

1 蛇型磁带及其读写特点

传统磁带是顺序读写设备,因为定位时间长,一直用于脱机文档存储和数据备份.最近几年,人们开始把磁带作为海量数据的联机存储设备,并研制出新的高速磁带——蛇型磁带[8].蛇型磁带沿着磁带的延长方向写数据.它首先沿着一个方向写一个磁道,写到磁道末尾,调转磁头方向,沿着另一个方向写下一个磁道.如此下去,直至所有磁道写完.在蛇型磁带中,磁头可以在磁道之间,即磁带的宽度方向上做径向移动.例如,当前磁头位置在第 0 磁道,要访问的目标地址在第 2 磁道上.传统的磁带技术需要磁带读写头顺序经过第 0 磁道、第 1 磁道,当到达第 2 个磁道时开始访问.在蛇型磁带中,磁带读写头直接从第 0 个磁道径向移动到第 2 个磁道,磁带读写头的定位距离大大减少.

2 蛇型磁带外排序算法

蛇型磁带外排序算法 STESort 的执行分为两个阶段.第 1 个阶段为内排序阶段,即将磁带上的数据按内存大小分段读入内存进行内存排序,形成多个有序的独立排序段,合理地分布在蛇型磁带上.第 2 个阶段为合并阶段,将蛇型磁带上的所有排序段合并成最终的有序数据文件.

2.1 内排序阶段

蛇型磁带的磁带表面有 S 个磁道(通常 $S=2^c$,c 为正整数).其中 $S/2$ 个磁道需沿着从磁带头到磁带尾的方向进行访问,称为正向磁道组;另外的 $S/2$ 个磁道需沿着磁带尾到磁带头的方向进行访问,称为反向磁道组.

在内排序阶段,将待排序磁带数据文件 F 按照内存缓冲区大小分段读入内存,进行内存排序,在硬盘缓冲区中形成多个局部有序的磁盘排序段 *diskrun*.当磁盘缓冲区写满以后,调用磁盘 2 路归并 diskmerge 程序,将磁盘缓冲区中所有的 *diskrun* 合并成一个磁带排序段 *taperun*,写到蛇型磁带的一个磁道上.重复这个过程,直到所有的数据都经过内排序形成有序段写回磁带为止.

内排序阶段算法如下:

F:待排序的磁带数据文件,其大小为 N,N 小于磁带容量$|T|$;

T_1:蛇型磁带,初始存放待排序文件 F;

T_2:蛇型磁带,初始为空白磁带;

K:代表 K-路归并中同时进行归并的有序段的个数;

M:可用内存空间大小;

D:磁盘缓冲空间, $D=\left\lceil \frac{N}{2K} \right\rceil \le \frac{|T|}{S}$;

m: $m=[N/M]$, n: $n=[N/D]$.

(1) *Track*=0;
(2) FOR i=1 TO m DO
(3) 从磁带 T_1 读 M 大小的数据块到内存,用 QuickSort 内排序算法进行排序;
(4) 排序结果形成 *diskrun*,写到磁盘缓冲区 D;
(5) IF 磁盘缓冲区 D 已满 THEN
(6) 调用 diskmerge 程序,将磁盘缓冲区中的所有 *diskrun* 合并成一个 *taperun*;
(7) 将 *taperun* 写入 T_2 的第 *Track* 磁道;
(8) *Track*=*Track*+1;
(9) ENDIF
(10) ENDFOR

其中,diskmerge 程序采用磁盘 2 路归并算法合并磁盘缓冲区中的所有有序段.算法中 K 的选取与数据量大小有关.当待排序数据少于一个磁道的容量时,取 K=1/2;否则,取 $K=S/2$.令磁盘缓冲区的大小 $D=\left\lceil \frac{N}{2K} \right\rceil$,使得待排序的数据平均分布在每个磁道上.当 K=1/2 时,经过内排序阶段,所有数据已经变成有序集合,无须执行合并过程中的 K-路归并.在磁盘缓冲区 D 中,当数据写入磁带 T_2 时, 空闲下来的磁盘空间立刻用于写入新的内排序结果.

2.2 合并阶段

B_{size} 为磁带上一个节的大小.在内排序以后,每个磁道上共有 $p=[D/B_{size}]$个节.在第 1 遍归并中,正向磁道组和反向磁道组中的磁带排序段分别进行 K-路归并.第 2 遍归并将两个 K-路归并结果进行 2 路归并,形成最终的有序文件.在合并过程中,为磁道组中的每个磁道设置一对磁盘缓冲区.在一个磁盘缓冲区的数据进行合并的同时,另一个磁盘缓冲区用于从磁带读取数据.从而,合并算法和磁带的读取操作可以流水线并行.

合并阶段算法描述如下:

(1) 读正向磁道组每个磁道的第 0 节数据到 K 个磁盘缓冲区;
(2) j=1; While $j \le p-1$ 采用 pipeline 方式执行如下两个代码段:
(3) (a) 合并排序段.
(4) 用“败者树”[9]归并算法对磁盘缓冲的数据进行 K-路归并,归并结果顺序地写入磁带.
(5) (b) 读磁带数据到磁盘缓冲区.
(6) 读正向磁道组每个磁道的第 j 节中的数据到 K 个磁盘缓冲区.j=j+1.
(7) ENDWHILE
(8) 重复(1)~(7),对反向磁道组中的磁带排序段进行 K-路归并;
(9) 对两个 K-路归并结果进行一次 2 路归并,形成最终的有序数据集合,写入磁带.

3 算法分析

3.1 STESort算法的理论分析

在分析磁带外排序算法时,主要考虑算法的 CPU 时间、磁带数据定位、磁带数据 I/O 和换带时间这几个方面.在分析过程中,用 N 表示待排序数据,它小于等于一个磁带容量;D 表示磁盘缓冲区,小于等于一个磁道容量;M表示内存缓冲区大小;T_1,T_2表示两个蛇型磁带,S_t为磁带定位速度,R_t为磁带读写速度;B_{size}表示磁带逻辑数据块大小.

3.1.1 STESort 算法的 CPU 时间

在 STESort 算法内排序阶段的步骤(3)中,M 大小的数据进行 QuickSort 排序的平均复杂度为 $O(M\log_2 M)$.步骤(3)总共执行 $\left\lceil \frac{N}{M} \right\rceil$ 次,则执行 QuickSort 的时间代价为 $O(N\log_2 M)$.步骤(6)将磁盘缓冲区的磁盘排序段进行 2 路归并,平均时间复杂度为 $O\left(D\log_2 \frac{D}{M}\right)$.步骤(6)共进行 $\frac{N}{D}$ 次,则执行 diskmerge 的时间代价为 $O\left(N\log_2 \frac{D}{M}\right)$.STESort 算法内排序阶段的 CPU 代价为 $O(N\log_2 D)$.

在 STESort 算法合并阶段,数据从磁带读入内存使用"败者树"算法[9]进行 K-路归并.在 K 个排序段中,每产生一个归并结果,都需要进行 $\log_2 K$ 次比较.正向磁道组的 K-路归并的比较次数为 $\frac{N}{2}\log_2 K$.同理,反向磁道组的 K-路归并的比较次数也为 $\frac{N}{2}\log_2 K$.则在第 1 遍 K-路归并中,CPU 代价为 $O(N\log_2 K)$.K-路归并结束后产生两个有序段,需进行一遍 2 路归并.2 路归并的 CPU 代价为 $O(N)$.STESort 算法合并阶段的 CPU 代价为 $O(N\log_2 K)$.综上所述,STESort 算法总的 CPU 时间为 $O(N\log_2 N)$.

3.1.2 STESort 算法的磁带定位

在蛇型磁带中,磁道之间存在一个很小的间隔.因为排序数据量很大,在计算定位时,可以忽略磁道之间作径向移动产生的微小距离.在内排序阶段,磁盘缓冲区的使用使磁带数据的读入为流状态,没有起停延迟,也没有定位操作.合并阶段一共进行了两遍归并.在第 1 遍归并中,读正向磁道组的一组数据到 K 个磁盘缓冲区,需要反向定位 K–1 次,定位距离为$(K-1)B_{\text{size}}$.读反向磁道上的一组数据到 K 个磁盘缓冲区的情况与此类似,需要正向定位距离$(K-1)B_{\text{size}}$.反向磁带定位和正向磁带定位在定位时间上存在差异.这里,为了简化讨论,忽略了这个微小差异.在第 1 遍归并中,定位距离总共为 $2(K-1)D$.第 2 遍归并为 2 路归并,定位距离为 $\frac{N}{2}$.在海量数据排序中,K 为常数,STESort 算法的磁带定位代价为 $O(N)$.

3.1.3 STESort 算法的磁带 I/O

在内排序阶段,所有磁带数据读、写各一次.在归并阶段,进行两遍归并,磁带所有数据的读、写各两次,因此,磁带 I/O 总量为 6N.STESort 算法的磁带 I/O 的代价为 $O(N)$.可以看到,STESort 算法的磁带 I/O 与待排序数据量呈线性关系,适合于对海量数据的排序操作.

3.1.4 SETSort 算法的换带时间

假定算法开始时,T_1,T_2 已经在磁带驱动器中.T_1 写有待排序的数据,T_2 为空磁带.算法先从 T_1 顺序读数据到内存进行内排序,并将磁带排序段写入 T_2.然后需要回绕这两条磁带,从 T_2 读数据,进行 K-路归并,结果写入 T_1.在第 2 遍归并之前,仍需回绕磁带.从 T_1 读数据进行 2 路归并,结果写入 T_2.在整个过程中,磁带需要回绕 2 次.换带时间代价为常数.

3.2 与蛇型磁带2路归并排序算法的比较

磁带是顺序读写的设备,实现外排序至少需要两个磁带:一个用于读,一个用于写.两个磁带驱动器的 2 路归并算法是传统磁带排序算法中的经典算法,也是最基本、最常用的算法[1].因此,本文采用传统的 2 路归并算法,在相同条件下与 STESort 算法进行分析和比较.

3.2.1 蛇型磁带 2 路归并算法

将传统的 2 路归并算法应用到蛇型磁带上,得到蛇型磁带 2 路归并算法.算法的执行过程描述如下:

(1) 根据内存 M 的大小,分段将磁带上的数据读到内存;

(2) Quicksort 内排序,将内排序产生的排序段写入磁盘,形成磁盘排序段;

(3) 在磁盘上将多个磁盘排序段合并成一个大的磁带排序段,写回磁带;

(4) 重复(1)~(3),直到所有的数据完成内排序为止;

(5) 重复进行 $\log_2(N/D)$ 次 2 路归并,产生最终的排序结果,保存在磁带上.

3.2.2 分析与对比

STESort 算法在 CPU 时间、磁带定位、磁带 I/O 和换带时间这 4 个方面,与蛇型磁带 2 路归并算法的理论分析结果见表 1.可以看到,传统的 2 路归并排序算法在数据量很大时,归并遍数增多.磁带定位、磁带 I/O 以及换带时间均随着迭代次数的增长而增长.STESort 算法的代价随着数据量呈线性增长,更适合于海量数据的排序.

Table 1 Theoretic analysis and comparison between two algorithms

表 1 两种算法的理论分析比较

Algorithms	CPU time	Tape locate	Tape I/O	Tape switch time
2-Way merge tape sort	$O(N\log_2 N)$	$O(N^2)$	$O\left(N\log_2\frac{N}{D}\right)$	$O(N^2)$
STESort	$O(N\log_2 N)$	$O(N)$	$O(N)$	Constant

4 模拟实验和实验结果

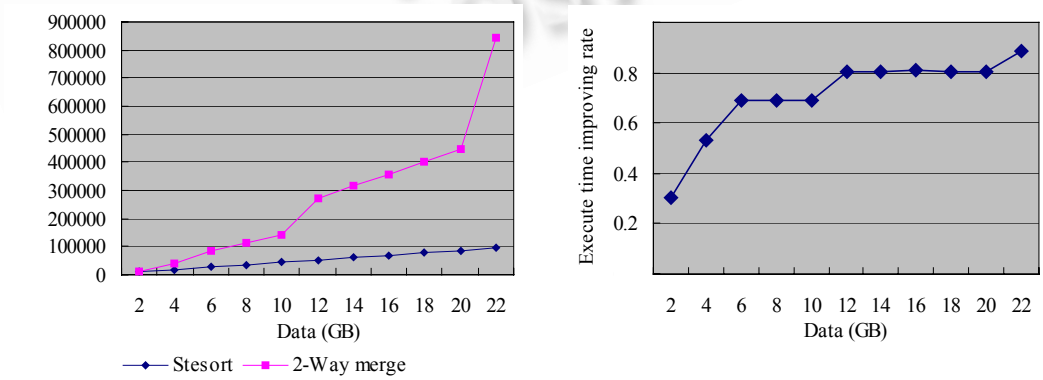
4.1 模拟实验环境

STESort 算法的实现环境为一个主频 850MHZ 的 Pentium III PC 机,操作系统为 Windows 2000 server,主存 256MB,80GB 硬盘.在实验中,用磁盘模拟磁带存储数据,使用 Hillyer 提出的 DLT4000 磁带定位时间模型[10]模拟磁带的定位和 I/O 操作.在 STESort 算法中,使用该模型模拟两个 DLT4000 磁带驱动器和两个容量为 20GB 的磁带.磁带的读写速率为 1.536MB/s,定位速度为 4.46MB/s.

4.2 实验结果

蛇型磁带 DLT4000 有 64 个磁道,排序数据量 N 在 2GB~22GB 之间变化.因为实验数据量足够大,取 K=32.磁盘缓冲区 D 取 $\left\lfloor\frac{N}{2K}\right\rfloor$,内存缓冲区 M 固定为 64KB.为了简化算法执行,实验中的记录只包含一个字段,即关键字段.该字段为整型值,满足均匀分布.每次实验执行 3 次,记录平均值.

图 1(a)是 STESort 算法与蛇型磁带 2 路归并算法在相同条件下的执行时间对比.内排序阶段的 QuickSort 算法的执行时间、磁盘上的排序段归并时间以及合并阶段的 K-路归并算法执行时间,是实验记录结果.磁带的定位时间通过 DLT4000 蛇型磁带的定位时间模型计算得到.数据量在 2GB~22GB 之间变化,每次增量为 2GB.从图 1(a)可以看到,STESort 算法的总体执行时间明显地优于蛇型磁带 2 路归并算法.随着数据量的增加,STESort 算法执行时间的增长速度比蛇型磁带 2 路归并算法缓慢.



(a) Execute time (a) 执行时间

(b) Execute time improving rate (b) 执行时间改善率

Fig.1 Comparison of execute time between two algorithms

图 1 两种算法执行时间比较

令 $$执行时间改善率=\frac{蛇型磁带2路归并算法执行时间-STESort算法执行时间}{蛇型磁带2路归并算法执行时间}\times 100\%,$$

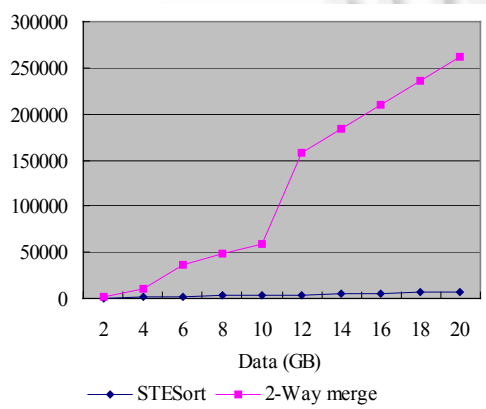
则算法执行时间改善率如图 1(b)所示.可以看出,随着数据量的增加,算法执行时间改善率逐渐增加.

定义磁带定位时间改善率和磁带 I/O 时间改善率如下:

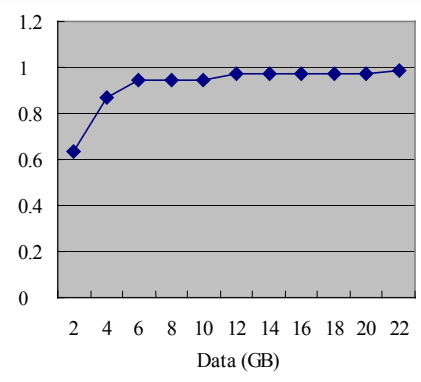
$$磁带定位时间改善率=\frac{蛇型磁带2路归并算法磁带定位时间-STESort算法磁带定位时间}{蛇型磁带2路归并算法磁带定位时间}\times 100\%,$$

$$磁带I/O时间改善率=\frac{蛇型磁带2路归并算法磁带I/O时间-STESort算法磁带I/O时间}{蛇型磁带2路归并算法磁带I/O时间}\times 100\%.$$

图 2(a)给出两种算法在磁带定位时间上的比较.图 2(b)是 STESort 算法相对于蛇型磁带 2 路归并算法的磁带定位时间改善率.图 3(a)是两种算法在磁带 I/O 时间上的比较.图 3(b)是 STESort 算法相对于蛇型磁带 2 路归并算法的磁带 I/O 时间改善率.可以看出,STESort 算法在磁带定位时间和磁带 I/O 时间上都优于蛇型磁带 2 路归并算法.随着数据量的增加,STESort 算法相对于蛇型磁带 2 路归并算法的磁带定位时间改善率和磁带 I/O 时间改善率也逐渐提高.此外,STESort 算法相对于蛇型磁带 2 路归并算法,磁带定位时间的改善比磁带 I/O 时间的改善更为明显.

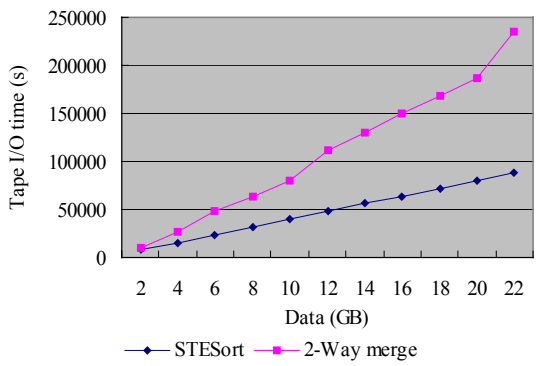


(a) Tape locate time
(a) 磁带定位时间

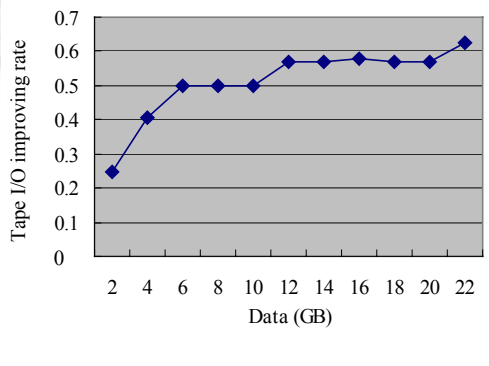


(b) Tape locate time improving rate
(b) 磁带定位时间改善率

Fig.2 Comparison of tape locate time between two algorithms
图 2 两种算法的磁带定位时间比较



(a) Tape I/O time
(a) 磁带 I/O 时间



(b) Tape I/O time improving rate
(b) 磁带 I/O 时间改善率

Fig.3 Comparison of tape I/O time between two algorithms
图 3 两种算法的磁带 I/O 时间比较

5 结 论

本文提出的算法充分利用了蛇型磁带的物理特点,使用两个磁带驱动器实现了 K 路归并,减少了磁带定位时间和磁带 I/O 时间,从而降低了排序算法的总执行时间.理论分析和实验结果表明,STESort 算法在磁带数据 I/O 和磁带定位时间上都明显优于传统的 2 路归并算法.尤其在磁带定位时间上,STESort 算法体现了更好的优越性.研究表明,蛇型磁带适用于存储和处理海量数据,STESort 算法适用于海量数据排序.

STESort 算法是单一磁带上的排序算法,为基于机器手臂磁带库的多磁带数据排序算法奠定了基础.在进行多磁带数据排序时,我们只需先对每个磁带上的数据应用 STESort 算法进行排序,然后将多个磁带的排序结果进行合并,得到最终的排序数据集合.

致谢 哈尔滨工业大学计算机科学与技术学院的张兆功博士研究生对本文提出了很多有益的建议,在此表示感谢.

References:

[1] Knuth D. The Art of Computer Programming, Vol 3: Sorting and Searching. 2nd ed., Redwood, CA: Addison Wesley, 1973. 503~657.

[2] Pang H, Carey MJ, Livny M. Memory-Adaptive external sorting. In: Agrawal R, Baker S, Bell DA, eds. Proceedings of the 19th International Conference on Very Large Data Bases. Dublin: Morgan Kaufmann, 1993. 618~629.

[3] Aggarwal A, Plaxton CG. Optimal parallel sorting in multi-level storage. In: Sleator DD, ed. Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms. New York: ACM Press, 1994. 659~668.

[4] Azar Y, Vishkin U. Tight comparison bounds on complexity of parallel sorting. SIAM Journal on Computing, 1987,16(3):458~464.

[5] DeWitt DJ, Naughton JF, Schneider DA. Parallel sorting on a shared-nothing architecture using probabilistic splitting. In: Rishe N, ed. Proceedings of the 1st International Conference on Parallel and Distributed systems. IEEE Computer Society, 1991. 280~291.

[6] Salzbergl B, Tsukerman A, Gray J, Stewart M, Uren S, Vaughan B. FastSort: a distributed single-input single-output external sort. In: Garcia-Molina H, ed. Proceedings of the International Conference on Management of Data. New York: ACM Press, 1990. 94~101.

[7] Bitton D, DeWitt DJ, Hsiao DK, Menon J. A taxonomy of parallel sorting. ACM Computing Surveys, 1984,16(3):287~318.

[8] Sandsta O. Improving the access time performance of serpentine. In: Acharya S, ed. Proceedings of the 15th International Conference on Data Engineering. Sydney: IEEE Computer Society, 1999. 542~551.

[9] Yan WM, Wu WM. Data Structure. 2nd ed., Beijing: Tsinghua University Press, 1995. 298~299 (in Chinese).

[10] Hillyer BK. On the modeling and performance characteristics of a serpentine tape. In: Gaither BD, ed. Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems. New York: ACM Press, 1996. 170~179.

附中文参考文献:

[9] 严蔚敏,吴伟民.数据结构.第 2 版,北京:清华大学出版社,1995.298~299.