作业 6

SA20225085 朱志儒

1、 已知有关系模式 R(A, B, C) 和 S(B, C, D)，每个属性都占 10 个字节，请估计下面的逻辑查询计划 T(U)，S(U)以及结果关系中每个属性的 V 值（假设满足“Containment of Value Sets”，并且选择条件中的值都在关系中存在）：

$$U=\pi_{AD}[(\sigma_{A=3 \wedge B=5} R) \bowtie S)]$$

相应的统计量如下：

T(R) = 100000, V(R,A) = 20, V(R,B) = 50, V(R,C) = 150

T(S) = 5000, V(S,B) = 100, V(S,C) = 200, V(S,D) = 30

解： 令 $\mathrm{H} = \sigma_{A=3 \wedge B=5} R$，则

$$\mathrm{T(H)} = \frac{\mathrm{T(R)}}{\mathrm{V(R,A)} \cdot V(R,B)} = \frac{100000}{20 \times 50} = 100$$
$$\mathrm{S(H)} = \mathrm{S(R)} = 10 \times 3 = 30 \ \ B$$
$$\mathrm{V(H,A)} = 1$$
$$\mathrm{V(H,B)} = 1$$
$$\mathrm{V(H,C)} \leq 100$$

令 $\mathrm{G} = \mathrm{H} \bowtie \mathrm{S}$，则

$$\mathrm{T(G)} = \frac{\mathrm{T(H)} \cdot T(S)}{\max\{V(H,B), V(S,B)\} \cdot \max\{V(H,C), V(S,C)\}} = \frac{100 \times 5000}{100 \times 200} = 25$$
$$\mathrm{S(G)} = 30 + 10 = 40 \ \ B$$
$$\mathrm{V(G,A)} = \mathrm{V(H,A)} = 1$$
$$V(G,B) = V(H,B) = 1$$
$$\mathrm{V}(G,C) \leq \min\{V(H,C), V(S,C), T(G)\} \leq 25$$
$$\mathrm{V(G,D)} \leq \min\{V(S,D), T(G)\} \leq 25$$

故

$$\mathrm{T(U)} = \mathrm{T}(G) = 25$$
$$\mathrm{S(U)} = 2 \times 10 = 20 \ \ \mathrm{B}$$
$$\mathrm{V(U,A)} = \mathrm{V}(G,A) = 1$$
$$\mathrm{V}(U,D) \leq \mathrm{V(G,D)} \leq 25$$

2、 固态硬盘（Solid State Drive，SSD）是一种基于闪存的新型存储器，它与传统磁盘的主要区别之一是：传统磁盘的读写操作的速度相同，而 SSD 的读速度远快于写速度。同时，SSD 的读速度要远高于磁盘，而写速度则比磁盘慢。现在我们想将传统的两阶段多路归并排序算法移植到 SSD 上。假设 SSD 上一次读块操作的时间是 t，一次写块操作的时间是 50t，磁盘上的读/写块时间是 30t。对于给定关系 R：

- R 包含 100000 个元组，即 T(R) = 100000.
- 一个磁盘块大小为 4000 bytes.
- R 的元组大小为 400 bytes，即 S(R) ＝ 400.
- 关系 R 在磁盘上非连续存放
- 排序字段的大小为 32 bytes.
- 记录指针的大小为 8 bytes.

现在我们考虑下面一种改进的归并排序算法。原来的两阶段归并排序的第一阶段是将排序后的整个元组写到 chunk 中，现在我们仅将排序后的<sortingKey, recordPointer>写出。第一阶段，我们在内存中将记录按<sortingKey, recordPointer>排序，当<sortingKey, recordPointer>记录填满内存时将其写到 chunk 中。第二阶段，读入各个 chunk 中的<sortingKey, recordPointer>并在内存中归并。通过记录指针(recordPointer)我们可以读取记录的其它部分(从 R 的磁盘块中)，并将排好序的记录写回到外存。请回答：

1） 如果 R 存储在磁盘上，这一改进排序算法的 I/O 代价（用 t 的表达式表示，包括最后写出到排序文件中的代价）是多少？并解释该算法性能是否能优于原来的排序算法。

2） 如果 R 存储在 SSD 上，这一改进排序算法的 I/O 代价（用 t 的表达式表示，包括最后写出到排序文件中的代价）是多少？并解释该算法性能是否能优于原来的排序算法。

解： 1）如果 R 存储在磁盘上，由于关系 R 在磁盘上非连续存放，将元组全部读入内存的时间为

$$100000 \times 30t = 3000000t$$

一个磁盘可以装< sortingKey, recordPointer >的数量为

$$4000 \div (32 + 8) = 100$$

将排序后的< sortingKey, recordPointer >全部写到磁盘的时间为

$$100000 \div 100 \times 30t = 30000t$$

将各个 chunk 中的< sortingKey, recordPointer >读入到内存的时间为

$$100000 \div 100 \times 30t = 30000t$$

通过记录指针(recordPointer)读取记录的其它部分到内存的时间为

$$100000 \times 30\mathrm{t} = 3000000\mathrm{t}$$

所有元组排序后写出到磁盘的时间为

$$100000 \div (4000 \div 400) \times 30t = 300000t$$

综上可知，改进排序算法的总时间代价为

$$3000000\mathrm{t} + 30000\mathrm{t} + 30000\mathrm{t} + 300000\mathrm{t} + 3000000\mathrm{t} = 6360000\mathrm{t}$$

而原来的排序算法的总时间代价为

$$100000 \times 30\mathrm{t} + 100000 \div (4000 \div 400) \times 30t \times 3 = 3900000t$$

显然 $6360000\mathrm{t} > 3900000\mathrm{t}$，故改进的排序算法没有优于原来的排序算法。

2）如果 R 存储在 SSD 上，将元组全部读入内存的时间为

$$100000 \times \mathrm{t} = 100000\mathrm{t}$$

将排序后的$<$ sortingKey, recordPointer $>$全部写到 SSD 的时间为

$$100000 \div 100 \times 50\mathrm{t} = 50000\mathrm{t}$$

将各个 chunk 中的$<$ sortingKey, recordPointer $>$读入到内存的时间为

$$100000 \div 100 \times \mathrm{t} = 1000\mathrm{t}$$

通过记录指针(recordPointer)读取记录的其它部分到内存的时间为

$$100000 \times \mathrm{t} = 100000\mathrm{t}$$

所有元组排序后写出到 SSD 的时间为

$$100000 \div (4000 \div 400) \times 50t = 500000t$$

综上可知，改进排序算法的总时间代价为

$$100000\mathrm{t} + 50000\mathrm{t} + 1000\mathrm{t} + 100000\mathrm{t} + 500000\mathrm{t} = 751000\mathrm{t}$$

而原来的排序算法的总时间代价为

$$\begin{aligned} &100000 \times \mathrm{t} + 100000 \div (4000 \div 400) \times 50t \times 2 + 100000 \div (4000 \div 400) \times t \\ &= 1110000t \end{aligned}$$

显然 $751000\mathrm{t} < 1110000\mathrm{t}$，故改进的排序算法优于原来的排序算法。

3、 我们在课本上讨论的归并排序算法是一个两趟算法。设两个连接关系为 R1 和 R2，在基于两趟归并排序的排序连接算法中，我们要求内存 M 必须满足条件 $M \geq \max\{\sqrt{R1}, \sqrt{R2}\}$。现在我们考查关系 R 的两趟归并排序算法，我们发现当内存 M 不满足条件 $M \geq \sqrt{R}$ 时，我们仍可以采用一种多趟算法来完成归并排序操作。请用自然语言或伪码给出这一多趟归并连接算法的简要描述和步骤，并给出当 B(R1)=10000，B(R2)=5000，M＝20 时该算法的 I/O 代价，这里我们假设 R1 和 R2 都是连续存放的。

解：多趟归并连接算法：

（1） 令 $T = B(R)$，从磁盘中读取 R 的 M 个块到内存的 Buffer 区，在每个块中采用内排序算法使得元组在块内有序，并且使得第 1 个块中元组的最小值小于第 2 个块的最小值，第 2 个块的最小值小于第 3 个块的最小值，以此类推，即前一个块的最小值小于后一个块的最小值，再将这 M 个块作为 1 个 chunk 写出到磁盘中，重复上述操作直到 R 中元组均为块内有序，chunk 中以块的最小值有序，进入（2）。

（2） 令 $T = \left\lceil \frac{T}{M} \right\rceil$，即现有 T 个 chunk，将该 T 个 chunk 作为一个集合 S，从 S 中选 M 个 chunk 出来作为 chunk 数组 H，集合 S 删除选中的 chunk。然后分别从 chunk 数组 H 的每个 chunk 中选第 1 个块组成块数组 K，并读入到内存的 Buffer 区，采用归并算法进行排序，将结果存入 1 个输出块中。

（a） 若块数组 K 的第 i 个块中元组全被选完，则从 chunk 数组 H 的第 i 个 chunk 中读取下一个块到内存中。

（b） 若输出块被填满，则将其写回到磁盘中。

本次操作得到的多个输出块组成一个新的 chunk，以便下次迭代使用。重复上述操作直到集合 S 为空，进入（3）。

（3） 若 $T > 1$，则重复（2）进行一次新迭代，直到 $T = 1$，即最后得到的 1 个 chunk 就是排好序的关系 R。

（4） 使用上述算法将 R1 和 R2 排序，最后将 R1 和 R2 归并连接。

IO 代价分析：

对于关系 R1，上述算法的迭代次数为

$$\lceil \log_{\mathrm{M}} B(R1) \rceil = \lceil \log_{20} 10000 \rceil = 4$$

IO 次数为

$$4 \times 10000 \times 2 = 80000$$

对于关系 R2，上述算法的迭代次数为

$$\lceil \log_M B(R2) \rceil = \lceil \log_{20} 5000 \rceil = 3$$

IO 次数为

$$3 \times 5000 \times 2 = 30000$$

将 R1 和 R2 归并连接的 IO 次数为

$$10000 + 5000 = 15000$$

综上可知，总的 IO 次数为

$$80000 + 30000 + 15000 = 125000$$