CS186 Section 1

Jan 26, 2016

Yifan

INTRO

please give feedback!

Note: this week it's mostly answers to worksheet, for future weeks I will add more class material summaries!

Materials to Cover via Worksheet

- External Sorting (Merge) & Internal Sort (heap)
- External Hashing
- Basic SQL Queries

What is time-space rendezvous and why do we use it?

- When records are in the same place (RAM) at the same time
- Useful when certain records need to be co-resident but are not guaranteed to be in the same input chunk
- Important in out-of-core algorithms (external sorting/hashing)

What is the difference between external sorting and external hashing?

- Sorting: conquer and merge, good if we need to sort or already sorted
- Hashing: divide and conquer, good if we need to remove duplicates
- Same memory requirement and IO cost (for 2 passes)

Why can we process B * (B - 1) pages of data with external hashing in just two passes (divide and conquer phases)?

- In Pass 1, there is 1 input buffer and B-1 output buffers.
- In Pass 2, we could hash B pages of data

If you're processing exactly B * (B - 1) pages of data, is it likely that you'll have to perform recursive external hashing? Why?

- Need perfectly even distributions

While you recursively perform external hashing, you reuse the same hash functions for partitioning. What's the problem with this?

The new partition wouldn't get any smaller!

List the differences between 2-way external merge sort and general external merge sort

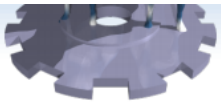
General External Merge Sort

☛ ***More than 3 buffer pages. How can we utilize them?***

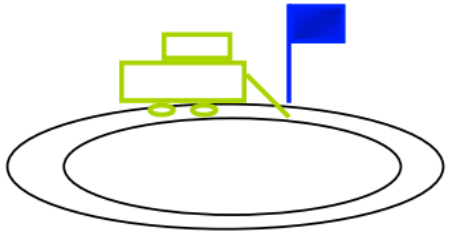
- Key Insight #1: We can merge more than 2 input buffers at a time… affects fanout → base of log!
- Key Insight #2: The output buffer is generated incrementally, so only one buffer page is needed for any size of run!
- To sort a file with N pages using B buffer pages:
  - Pass 0: use B buffer pages. Produce $\lceil N / B \rceil$ sorted runs of B pages each.
  - Pass 2, …, etc.: merge B-1 runs, leaving one page for output.

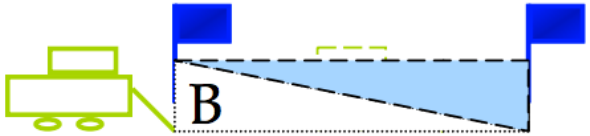
N	B=3	B=5	B=9	B=17	B=129	B=257
100	7	4	3	2	1	1
1,000	10	5	4	3	2	2
10,000	13	7	5	4	2	2
100,000	17	9	6	5	3	3
1,000,000	20	10	7	5	3	3
10,000,000	23	12	8	6	4	3
100,000,000	26	14	9	7	4	4
1,000,000,000	30	15	10	8	5	4

More on Replacement Sort



- Fact: average length of a run is *2B*
- The "snowplow" analogy
  - Imagine a snowplow moving around a circular track on which snow falls at a steady rate.
  - At any instant, there is a certain amount of snow S on the track. Some falling snow comes in front of the plow, some behind.





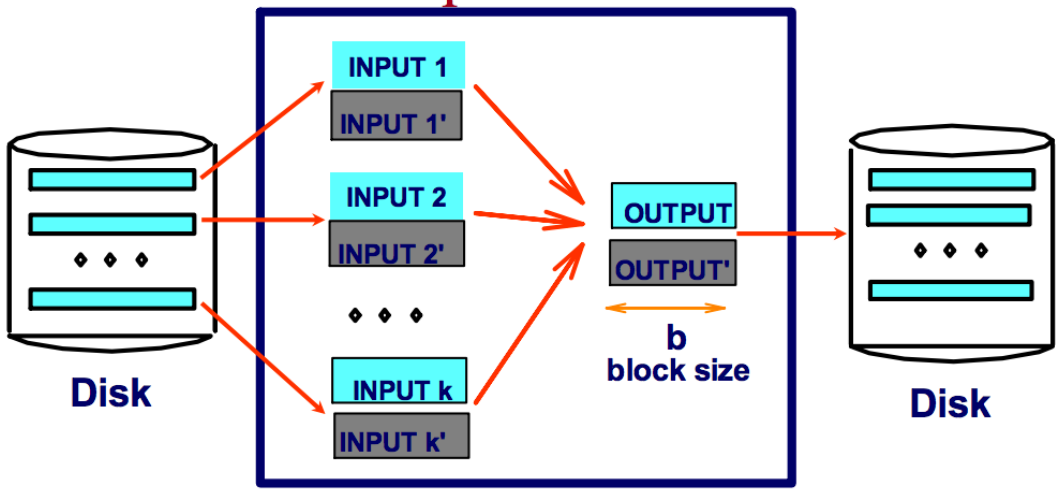
  - During the next revolution of the plow, all of this is removed, plus 1/2 of what falls during that revolution.
  - Thus, the plow removes 2S amount of snow.

http://www.csbio.unc.edu/mcmillan/Media/Comp521F10Lecture17.pdf

Double Buffering

To reduce wait time for I/O request to complete, can *prefetch* into `*shadow block*'.

- Potentially, more passes; in practice, most files still sorted in 2-3 passes.



B main memory buffers, k-way merge

https://www.andrew.cmu.edu/course/15-415-f11/applications/ln/lecture12.pdf

SQL

```
SELECT [DISTINCT] <column expression list>
  FROM <single table>
[WHERE <predicate>]
[GROUP BY <column list>
 [HAVING <predicate>] ]
[ORDER BY <column list>];
```

SQL

Songs (song_id, song_name, album_id, weeks_in_top_40)

Artists (artist_id, artist_name, first_year_active)

Albums (album_id, album_name, artist_id, year_released, genre)

Find the 5 songs that spent the most weeks in the top 40, ordered from most to least.

```
SELECT song_name FROM Songs ORDER BY
weeks_in_top_40 DESC LIMIT 5;
```

Find the name and first year active of every artist whose name starts with the letter ‘B’

SELECT artist_name, first_year_active FROM Artists
WHERE artist_name LIKE ‘B%’;

(% is a wildcard in SQL).

Find the total number of “Techno” albums released each year

```
SELECT year_released, COUNT(*) FROM Albums
WHERE genre = ‘Techno’ GROUP BY year_released;
```

Find the genre and the number of albums released per genre; don't include genres that have a count of less than 10

```
SELECT genre, COUNT(*) FROM Albums GROUP BY genre HAVING COUNT(*) >= 10;
```

?

My OH: Thur & Fri 4 pm Soda 611