2.4 Records and Buckets

Blocks: the minimal units of data transfer from the disk to memory or vice versa.

Records: units of info, such as:
- all the info about one patient in a hospital
- all the info about one taxpayer

File: a collection of records of a given type.

Assumptions

- Files are stored "nearly contiguously" on the disk in a small number of large areas called **extents**.
  - (this may yield inaccurate estimates if the files are small, and they fit in one extent.)
    - T (extent read) >>> T (extent-to-extent seek)
  - Sequential reading:

    $b * ebt$
  - Random reading, one block at a time:

    $b * (s + r + btt)$
- We use only fixed-length record files (i.e., all records are of the same size)
  - (What if not? i.e., variable-length record files.)
- We have a single-user system.

Purpose

- To be able to estimate time for file operations
  - helps us to decide what file organization to use
- Remarks:
  - We should know how the file will be used beforehand
  - Even under simplifying assumptions we can make good enough estimations.

The Blocking Factor (*Bfr*)

Two cases:
- If R evenly divides B (no empty space in file)

  $Bfr = B / R$

  (number of bytes in a record)

  (number of bytes in a block)
- If R does not evenly divide B (empty space)

  $Bfr = floor (B / R)$

- Assume number of records in a file is n:

  $b = n / Bfr$

  number of blocks in the file
- What is the time to read the file sequentially from beginning to end ?

  $b * ebt = (n / Bfr) * ebt$
- Correlation with block size?
  - For a fixed B and R, as block size gets smaller (less records stored per block), what happens ?
  - The sequential reading time gets longer.

Example: Night School Transcripts

Question: How record placement affects sequential reading time?
- 1600 byte transcript records are stored for 30,000 students.
- Block size (B) is 2400 bytes

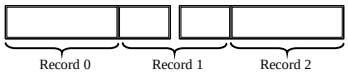
Examine the two cases:
- where each record is kept in a single block and empty space is left.
- Where each record spans two blocks, and all space is used.

Comment on sequential reading time?
- Sequential reading time depends on the amount of space the file takes up. When we have empty space, we have slower sequential reading.

We can save both time cost and space cost by choosing the appropriate file organization.

Choosing Bucket or Buffer Size

- We want to find one transcript record
  - We overlap block boundaries.
  - We need lookup tables/indexes which tell us the address of the record on disk.



- Two-block buckets: if we always read in two blocks and write out two blocks.
- **Bucket:** a logical collection of blocks on disk
- **Buffer:** such a collection of blocks in memory

Buckets

Bucket: a logical unit of data transfer for a file organization. A bucket consists of several consecutive blocks on disk.

- Block size on the disk == page size in OS
- Bucket size on disk == buffer size in memory

Effects of bucket size in sequential reading ops:

- Transcript Example.
- Once the record size evenly (or nearly evenly) divides the bucket size, so that no (little) space is wasted,
  - Efficient sequential reading is ensured.
  - Larger bucket sizes do not affect sequential reading time - same space utilization. Why? Because b is the same.

Usually we choose bucket size to optimize operations other than sequential reading.

How Random Reading Is Affected by Bucket Size ?

- Random reading of the entire file is a frequent op.
  - an index that indicates in what order to read the buckets is used
  - Reading order is not the same as records are stored in the disk.
- Random reading of the buckets in a file will be faster with larger bucket size.
  - The number of seeks will be smaller, and the total amount of data transfer time will be the same.
    - ***bk***: number of buckets (and assume ***bk*** = ***b***)
    - ***dtt***: data transfer time - time to transfer one bucket

Random time for reading all buckets = bk * (s + r + dtt)

Random time for reading of the records of a file = n * (r + s + dtt)

Bucket Size for Cambridge Tax File

Question: How doubling the bucket size from 2400B to 4800B affect various ops?

- 30,000 records of 100 bytes each.

- **Sequential reading:**
  - Calculate the average amount of tax being paid in Cambridge.
  - » Bucket size does not affect sequential reading time.
- **Random reading by records**
  - An index keeps a list of street names and the bucket addresses of tax records for taxpayers on that street.
  - » Larger bucket size makes random reading less efficient.

Optimal Bucket Size for a Mix of Operations

Question:

- In a typical month we make 1000 individual record fetches in the Cambridge tax file.
  - Lookup table is brought into memory in 1 disk access
- In addition we read the whole file randomly by bucket, using a list of bucket addresses kept in memory.

What is the optimal bucket size for this mix of operations?

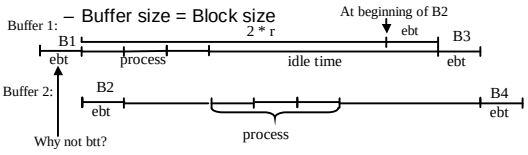
2.5 Double Buffering

What if the application requires to read buckets one by one and make calculations on each bucket?

- The problem is solved by having two buffers (double-buffering) for reading:
  - Each buffer is an array of bytes in memory that has the same size as one bucket.
- For short & simple calculations, double buffering can reduce the cost in CPU time to nearly zero, because it is always overlapping I/O time.
  - Ex: finding the average salary of all employees in an employee file (i.e., a file of employee records.)

When Double Buffering Does Not Work

- Sometimes we cannot process the data fast enough to use double buffering:
  - When computations are too complex
  - Ex: calculations take three times the read time for the data (2.52 ms to process data in one 2400-byte block) (btt=0.8 ms, ebt=0.84 and r=8.3 ms)
  - Buffer size = Block size



Cost of reading and processing the whole file = ?

- What if we used track-sized buffers?
  - Figure 2.6
  - If we use track-sized buffers, and x is the multiple of I/O time needed for calculations,

Total time = ceiling(x) * b * ebt

For our case:

Total time = 3 * b * ebt

Assumptions

- Double-buffering is possible most of the time.
  - Only time to read in and write out data counts
  - CPU time is overlapped with i/o time, hence it has 0 cost.

In the rest of the course, we will use track-sized buffers to do double-buffering.

Modifying All Records in a File

- First ,suppose we want to modify one record, and that the bucket size is one block:

Time to find and modify a block=s + r + btt + 2r

- Time required to modify all records in a file?
  - Ex: Add a surtax of 5% on the Cambridge taxpayers.
  - If Double buffer size = b ?
  - If Double buffer size = 1 track ?
    - 20 blocks of 2400 bytes on each track.
    - The track is traversed exactly twice: once for R,once for W
    - Records are replaced in exactly the same place they came from.
    - Total time = ?