764 Note

Sorting, concurrency control, logging, recovery, and two-phase commit( i.e., correct termination of distributed transactions). (Storage structures, the topic we started today, will be in the final but not the midterm.)

The topics will include two-phase commit, i.e., correct termination of distributed transactions.

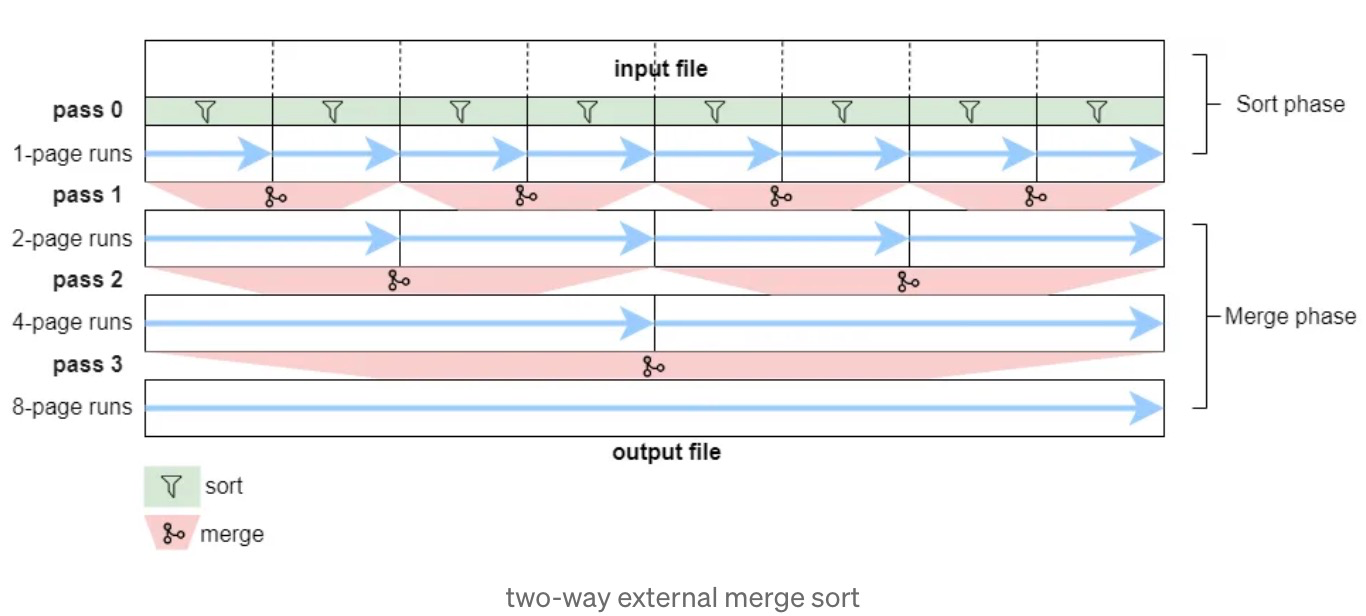
20道题目

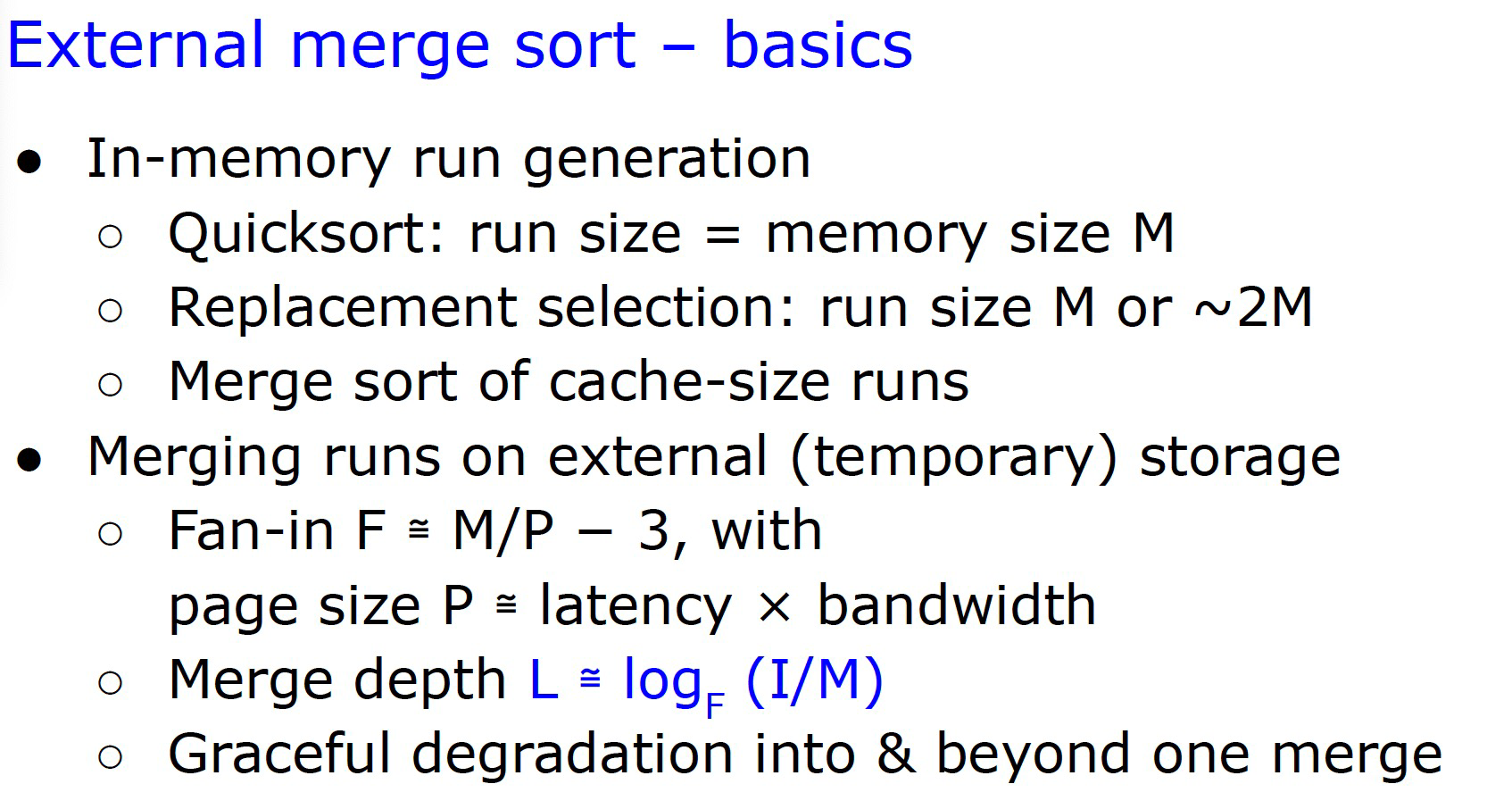
Sorting

cannot beat O(N log N) complexity

有机会的话需要画图

External merge sort





为什么quick sort适合internal sort

 no need additional storage space

ACID

atomicity, consistency, isolation, and durability

脏读 (Dirty Read). 不可重复的读 (Unrepeatable Read) 幻象读 (Phantom Read).和隔离级别这篇说的最好:

<https://www.51cto.com/article/697668.html>

SI is repeatable Read 隔离级别

MVCC without lock use snapshot.

2PL Deadlock is also possible in 2-PL + can release locks before commits+not insure recoverabilit+不能解决脏读

Strict Two Phase Locking  release the locks only when commits.

Rigorous 2PL，The data written by each transaction cannot be read or rewritten by other transactions before it concludes.

Early Lock Release:

* txn **releases** all locks without waiting for the commit record to be written to disk.
* By releasing its locks before it commits, other transactions can read the pre-committed transaction’s dirty data. Call these dependent transactions.

Controlled lock violation:

retains the locks until transaction end, not release,allow conflict

Controlled lock violation VS early lock release

Controlled lock violation is also quite different from early lock release. First, transaction T0 retains its locks in controlled lock violation as long as in traditional lock release, so it does not release any locks early. Second, locks can be violated while trans-action T0 commits, but only after T0 has added its commit record to the log buffer. Third, hardly any new mechanism is required – for example, neither the allocation nor the clean up of tags are required. Fourth, if there is no conflict between committing and active transactions, controlled lock violation does not impose any restrictions or overheads, whereas early lock release forces com-mitting transactions to install tags in the lock manager’s hash table just in case a subsequent transaction might create a lock conflict. Finally, and perhaps most importantly, controlled lock violation requires very little if any new theory – violation of lock conflicts and resulting commit dependencies are already part of the traditional theory of concurrency control and recovery.

Why use Controlled lock violation?

1. simple
2. precise
3. well work with 2PL

Controlled lock violation and early lock release during hardening

false conflicts due to lock sizes, the granularity of the locks is too coarse

**Deferred lock acquisition and deferred lock enforcement**

these two execution of the application logic.