CS634
Lecture 8, Feb 24, 2016

Managing Disk Resources, cont.

These slides are not based on "Database Management Systems" 3rd ed, Ramakrishnan and Gehrke

Disk Resource: Find out about disks on topcat

- One way: read the system log as the system comes up
- Linux: the **`dmesg`** tool outputs the system log:

```
…
[ 1.220567] ata4.00: ATA-7: WDC WD2500JS-75NCB3, 10.02E04, max UDMA/133
[ 1.220654] ata3.00: ATA-7: WDC WD2500JS-75NCB3, 10.02E04, max UDMA/133
[ 1.228713] scsi 2:0:0:0: Direct-Access     ATA      WDC WD2500JS-75N 10.0 PQ: 0
ANSI: 5
[ 1.229069] scsi 3:0:0:0: Direct-Access     ATA      WDC WD2500JS-75N 10.0 PQ: 0
ANSI: 5
…
```

- This show two Western Digital disks (do a web search on `WDC WD2500JS-75NCB3` to find out about them)
- It's a SATA disk, but that technology is under the SCSI umbrella in the kernel. SCSI is the older technology, and dbs2's disks are actual SCSI disks.
- Another way: look at /proc/scsi/scsi, a pseudofile maintained by the kernel, find same disk description
- We also found the system has 4GB of memory from dmesg and /proc/meminfo.

Partitions in use for filesystems on topcat's disks

Use "df –l" to see local filesystems, look for /dev/sd* entries:

```
topcat$ df -l
Filesystem     1K-blocks    Used Available Use% Mounted on
udev             2013052       4   2013048   1% /dev
tmpfs             404784     464    404320   1% /run
/dev/sda1      236049160 6643040 217392452   3% /
… (no more /dev/… entries)
```

This shows only one disk in use for filesystems, /dev/sda, and only one partition of it, /dev/sda1.

Partitions on topcat

List all partitions, whether in active use or not:

```
eoneil@topcat:~/634$ sudo lsblk -o NAME,FSTYPE,SIZE,MOUNTPOINT,LABEL
[sudo] password for eoneil:
NAME   FSTYPE   SIZE MOUNTPOINT LABEL
sda           232.9G
├─sda1 ext4   228.9G /
├─sda2            1K
└─sda5 swap       4G [SWAP]
sdb           232.9G
└─sdb1 ext2   232.8G
sr0            1024M
```

- This shows both disks, /dev/sda and /dev/sdb, and the fact that /dev/sda has two partitions in use:
  - /dev/sda1 for the root filesystem (i.e. the whole local filesystem in this case)
  - /dev/sda5 for swap space, used by the kernel for virtual memory pages, etc.
- So /dev/sdb is apparently healthy but not in use on this system

MySQL 5.1-5.6/Innodb: just one tablespace

- Global variable (can be set in /etc/mysql/my.cnf)
- *innodb_data_home_path* gives list of files making up the one tablespace:
- The default is:
  - innodb_data_file_path = ibdata1:10M:autoextend
  - Here ibdata1 is the filename relative to *innodb_data_home_dir* (if non-null) or *datadir*
  - This starts with one 10M file and lets it grow by 8M extensions

Topcat's mysql (v. 5.6) has datadir = /var/lib/mysql and no setting for innodb_data_home_dir or innodb_data_file_path
  - So our data file is /var/lib/mysql/ibdata1.
- MySQL docs: System globals index

MySQL Data files

- topcat$ sudo ls -l /var/lib/mysql

```
…
drwx------ 2 mysql mysql     4096 May 14  2015 huang001db
drwx------ 2 mysql mysql     4096 Nov 25 08:24 hwangdb
-rw-rw---- 1 mysql mysql 44040192 Feb 23 06:39 ibdata1
-rw-rw---- 1 mysql mysql 50331648 Feb 23 06:39 ib_logfile0
-rw-rw---- 1 mysql mysql 50331648 Feb  8 02:41 ib_logfile1
drwx------ 2 mysql mysql     4096 May 15  2015 indusdb
…
```

- Here ibdata1 is the data file, and we see two redo log files too.
- Each database has a small directory here, but all the table and index pages are in the one big file.

Adding a file to the MySQL tablespace

- To add /disk2/ibdata2 as a file, we freeze the current datafile by removing autoextend, specify it with a full path, and add the new file to the list, with autoextend if desired:
  innodb_data_file_path =
  /var/lib/mysql/ibdata1:36M;/disk2/ibdata2:50M:autoextend
- Specifically, we bring the server down, change my.cnf, and bring the server up again.
- See MySQL 5.6 manual, sec. 14.4.1.
- These added file paths may specify software or hardware RAID, or even raw partitions.

Living with a single tablespace

- With two projects sharing one server, it is common to run two instances of mysql, one for each project.
- Then each project has its own defined disks
- DBA must add up needed CPU and memory resource needs
- Or be an early adopter of version 5.7…
- Or just buy another server…

Innodb Log files

- The redo log file location is *innodb_log_group_home_dir* in my.cnf, or the datadir if this isn't set (topcat case).
- The undo log is in the main tablespace.
- If you do not specify any InnoDB log variables in my.cnf, the default is to create two redo log files named ib_logfile0 and ib_logfile1 in the MySQL data directory.
- To change the redo log location, say onto a mirrored RAID, bring down the server, which "uses up" the logs in a sense, edit the location, and bring up the server.
- Best to do this sort of setup as part of database initialization.

Basic Memory Config in MySQL

- Innodb has a buffer pool, size *innodb_buffer_pool_size*
- The size in bytes of the memory buffer InnoDB uses to cache data and indexes of its tables. The default value is 128MB (v 5.1-5.6, and in use on topcat).
- On a dedicated database server, you may set this to up to 80% of the machine physical memory size.
  - But of course not larger than the DB data itself.
  - Also raise *innodb_log_file_size* so total of log file sizes is *innodb_buffer_pool_size (Ref: see innodb_log_file_size docs)*
- See mysql manual (v 5.6) sec. 8.10.1 for more on the buffer pool
- With two mysql instances, make that <= 40% each.
- Note: quick way to find physical memory size on a UNIX/Linux system: use the "top" utility.

Basic Memory Config in Oracle (v 10)

- Two kinds of memory:
- SGA System Global area, including the database buffer caches
- PGA Program Global area, including memory for sorts, hashing, bitmaps, and bulk loads
- Oracle offers Automatic Shared Memory Management.
  - This has two major settable parameters, for the SGA and PGA areas, called SGA_TARGET and PGA_AGGREGATE_TARGET.
- On a dedicated database server, you may set the sum of these to up to 80% of the machine physical memory size.
- Could be .6*Mem for SGA, .2*Mem for PGA for example
- Once we get use of dbs3, revisit this for Oracle v. 11.

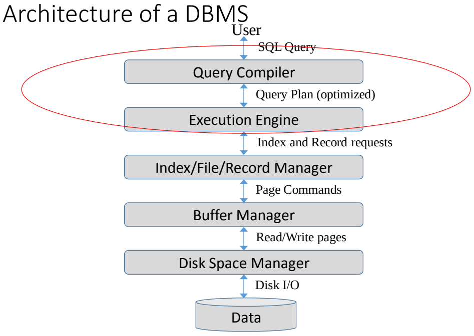
Oracle Memory Config

- If lots of complex queries or sorting, up the PGA size
- Note that Burleson notes problems with AMM, thinks a good DBA should take over memory tuning
- But that involves a lot of parameters.
- Most important is overriding ridiculously small default memory sizing

Query Evaluation Overview

Slides based on “Database Management Systems” 3rd ed, Ramakrishnan and Gehrke

Architecture of a DBMS



A first course in database systems, 3rd ed, Ullman and Widom

The two major parts of the DB engine

- QP = query processor, top two boxes on last slide
- Storage manager = rest of boxes
- See “index and record requests” flowing between
- Can be more specific, see list, pg. 283:
- Actions on “files”: file scan, search with equality selection, search with range selection, insert record, delete record
- Files listed: heap files, sorted files, clustered files, heap file with unclustered tree index, heap file with unclustered hash index.
- An index on its own is a sorted file.

Where are the tables?

- A table could be held in a heap file with multiple indexes. A file only has at most one currently relevant index, the one in use.
- The database can use multiple indexes for a single query, but that would mean the QP first working with (say) two indexes and then working with the results from that plus the table data file.
- So a file can be a simplification of a table (ignoring all but one of its indexes, or all of them) or an index itself
- The API can be called an ISAM (in one of its meanings), indexed sequential access method, allowing scans and lookup, range scan if tree-based.

Storage Engine API

- If a QP and storage engine hue to an API, then different storage engines can be “plugged in” to the database
- Example: MS SQL Server can access Excel files via the OLE-DB API. Also via ODBC.
  - That is, there is an Excel OLE-DB “provider” (you don’t need the whole Excel GUI).
- Example: MySQL has various storage engines—MyISAM and Innodb, etc.
  - New one (Nov ‘12): ClouSE uses Amazon S3 cloud storage. But seems to be a dead project now. S3 is actively used for backup of mysql data. Can’t just put the mysql datafile on S3.

Query Evaluation Overview

- SQL query first translated to relational algebra (RA)
  - Tree of RA operators, with choice of algorithm among available implementations for each operator
- Main issues in query optimization
  - For a given query, what plans are considered?
  - Algorithm to search plan space for cheapest (estimated) plan
  - How is the cost of a plan estimated?
- Objective
  - Ideally: Find best plan
  - Practically: Avoid worst plans!
- We will study the System R approach

Example Schema

Sailors

<u>sid</u>	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Boats

<u>bid</u>	name	color
101	interlake	red
103	clipper	green

Reserves

<u>sid</u>	<u>bid</u>	<u>day</u>
22	101	10/10/96
58	103	11/12/96

Example Query

Sailors (<u>*sid*: integer</u>, *sname*: string, *rating*: integer, *age*: real)
Reserves (<u>*sid*: integer, *bid*: integer, *day*: dates</u>)
Boats(<u>*bid*: integer</u>, *name*: string, *color*: string)

- *Find names of sailors who have rating 10 and who reserved a red boat.*

select sname from sailors s, reserves r, boats b
where s.sid = r.sid and r.bid = b.bid -- join conditions
and s.rating = 10 and b.color = 'red'
RA: on board: see joins, selection, projection operators

Statistics and Catalogs

- To choose an efficient plan, the QP needs information about the relations and indexes involved

- ***Catalogs*** contain information such as:
  - Tuple count (NTuples) and page count (NPages) for each relation
  - Distinct key value count (NKeys) for each index, INPages
  - Index height, low/high key values (Low/High) for each tree index
  - Histograms of the values in some fields (optional)

- Catalogs updated periodically
  - Updating statistics when data change too expensive
  - Approximate information used, slight inconsistency is ok
  - Both Oracle and mysql have an "analyze table" command for gathering stats, for use after a table load or massive update.
  - Both Oracle and mysql (v. 5.6+) store stats across DB shutdown/startup and automatically update them periodically.

Methods for Relational Operator Evaluation

Techniques:

- Indexing
  - Choose an index based on conditions in WHERE clause or join conditions
- Scan or Iteration
  - Reading a file entirely: file can refer to both data records file or index file
- Partitioning
  - Partition the input tuples and replace an expensive operation by similar operations on smaller inputs

Access Paths

- An access path is a method of retrieving tuples:
  - File scan
  - Index scan using an index that matches a condition
- A tree index matches (a conjunction of) terms that involve every attribute in a prefix of the search key
  - E.g., tree index on <a, b, c> matches the selection a=5 AND b=3, and a=5 AND b>6, but not b=3
- A hash index matches (a conjunction of) terms attribute = value for every attribute in the search key of the index
  - E.g., hash index on <a, b, c> matches a=5 AND b=3 AND c=5
  - but it does not match b=3, or a=5 AND b=3

Example of matching indexes

Pg. 399: fix error Sailors→ Reserves on line 8

Reserves (<u>*sid*: integer, *bid*: integer, *day*: dates</u>, *rname*: string) ← rname column added here

with indexes:

- Index1: Hash index on (rname, bid, sid)
  - Matches: rname='Joe' and bid = 5 and sid=3
  - Doesn't match: rname='Joe' and bid = 5
- Index2: Tree index on (rname, bid, sid)
  - Matches: rname='Joe' and bid = 5 and sid=3
  - Matches: rname='Joe' and bid = 5, also rname = 'Joe'
  - Doesn't match: bid = 5
- Index3: Tree index on (rname)
- Index4: Hash index on (rname)
  - These two match any conjunct with rname='Joe' in it

Executing Selections

- Find the *most selective access path*, retrieve tuples using it
  - Then, apply any remaining terms that don't match the index
- *Most selective access path:* index or file scan estimated to require the fewest page I/Os
  - Consider *day<8/9/94 AND bid=5 AND sid=3*
- If we have B+ tree index on *day*, use that access path
  - Then, *bid=5* and *sid=3* must be checked for each retrieved tuple
  - *day* condition is primary conjunct
- Alternatively, use hash index on *<bid, sid>* first
  - Then*, day<8/9/94* must then be checked

Example Schema

Sailors (<u>*sid*: integer</u>, *sname*: string, *rating*: integer, *age*: real)
Reserves (<u>*sid*: integer, *bid*: integer, *day*: date</u>, *rname*: string)

- Similar to old schema; *rname* added–name for the reservation itself
- Makes reserves even more clearly an entity (recall earlier discussion in class 2)
- Reserves:
  - 40 bytes long tuple, 100K records, 100 tuples per page, 1000 pages
- Sailors:
  - 50 bytes long tuple, 40K tuples, 80 tuples per page, 500 pages

Using an Index for Selections

- Cost influenced by:
  - Number of qualifying tuples
  - Whether the index is clustered or not
  - Cost of finding qualifying data entries is typically small
  - E.g.,

```
SELECT  *
FROM    Reserves R
WHERE   R.rname < 'C%'
```

- Assuming uniform distribution of names, 10% of tuples qualify, that is 10000 tuples
  - With a clustered index, cost is little more 100 I/Os
  - If not clustered, up to10K I/Os!

Executing Projections

- Expensive part is removing duplicates
  - DBMS don't remove duplicates unless DISTINCT is specified

```
SELECT  DISTINCT R.sid, R.bid
FROM    Reserves R
```

- Sorting Approach
  - Sort on <sid, bid> and remove duplicates
  - Cheap if an index with both R.sid and R.bid in the search key exists
- Hashing Approach
  - Hash on <sid, bid> to create partitions
  - Load partitions into memory one at a time, build in-memory hash structure, and eliminate duplicates

Executing Joins: Index Nested Loops

```
foreach tuple r in R do
    foreach tuple s in S where r_i == s_j do
        add <r, s> to result
```

- Cost = M + (M*p_R) * (cost of finding matching S tuples)
- M = number of pages of R, p_R = number of R tuples per page
- If relation has index on join attribute, make it inner relation
  - For each outer tuple, cost of probing inner index is 1.2 for hash index, 2-4 for B+, plus cost to retrieve matching S tuples
  - Clustered index typically single I/O
  - Unclustered index 1 I/O per matching S tuple