Transaction performance

Transaction performance relates among other things from IO performance means hard disk performance.

Hard disk performance

When you select a hard disk, an important feature to consider is the performance (speed) of the drive. Hard disks come in a wide range of performance capabilities. As is true of many things, one of the best indicators of a drive's relative performance is its price. An old saying from the automobile-racing industry is appropriate here: "Speed costs money. How fast do you want to go?"

The performance of a hard disk depends on several delays associated with reading or writing data on a computer's disk drive.

A measurement, called average access time (AAT), involves the elements, average seek time (AST), average rotational latency (ARL) and transfer time (TT).

```
average access time (AAT) = average seek time (AST)
                          + average rotational latency (ARL)
                          + transfer time (TT)
```

In a modern CPU, a millisecond is an enormous amount of time. A 1 GHz processor can theoretically execute over one million instructions in a millisecond! Obviously, even small reductions in AAT can result in improvements in overall system performance, because the rest of the system is often sitting and waiting for the hard disk during this time.

Average seek time (AST)

AST is normally measured in milliseconds (ms). It is the average amount of time it takes to move the heads from one cylinder (track) to another cylinder a random distance away. Seek time for a given disk varies depending on how far the head's destination is from its origin at the time of each read or write instruction.

Typical average seek time for a hard disk is about 8 ms and 4 ms for enterprise class high end disk drives. Modern PC hard disks have average seek times of 8 ms to 12 ms. Floppy disk drives and optical drives have much longer average seek times (e.g. floppy drive 100 - 200 ms AST, CD-ROM drive 120 - 400 ms). Solid state media can have average seek times of under 0.1 ms because there are no moving parts to align.

Small differences in seek time make a big difference. 6 ms and 9 ms don't sound that different, but 9 ms is 150% of 6 ms. Doing things that require constant seeking can take up to 50% longer on the 9 ms drive.

Average rotation latency (ARL)

ARL is the average time (in milliseconds) that it takes for a sector to be available after the heads have reached a track. On average, this figure is half the time that it takes for the disk to rotate one time, which is 4.17 ms at 7200 rpm. A drive that spins twice as fast would have half the latency.

ARL is the major component of AAL. Let's say you want to read some data that the read head just passed over. You have to wait for the entire disk to rotate once to read it. Assume 7'200 rpm = 120 rotations per second = 8.33 ms per rotation. That's a 8.33 ms wait worst case, and a 4.17 ms wait on average. That's why you don't often see seek times below 5 ms unless it's a 10'000 rpm drive.

Transfer time (TT)

The TT depends on the transfer rate. The transfer rate is the rate at which the drive and controller can send data to the system. The transfer rate depends primarily on the drive's HDA and secondarily on the controller. Transfer rate used to be more bound to the limits of the controller, meaning that drives that were connected to newer controllers often outperformed those connected to older controllers.

Example

The AST and ARL can be taken from the data sheet below. The TT can be calculated as follows with e.g. 16 kB InnoDB data block size:

```
TT = amount of data to transfer / max. media transfer rate
   = 16kB / 96.4 MB/s
   = 0.17 ms
```

Because this value (TT) is small compared to AST and ARL it is often neglected.

```
# cat /proc/ide/ide0/hda/model
HDS722580VLAT20

==> Google ==> data sheet
http://www.hitachigst.com/hdd/support/d7k250/d7k250.htm

Rotational speed               7200    rpm
Latency average                   4.17 ms      (= 60 / 7200 / 2, ARL)
max. Media transfer rate        757    Mbits/s (= 96.4 MB/s)
max. Interface transfer rate    100    MB/s
Average seek time               8.8    ms      (AST)
Seek time Track to track        1.1    ms
Seek time Full track           15.1    ms

AAT = AST + ARL + TT
    = 8.8 + 4.17 + 0.17
    = 13.1 ms
```

This gives us a theoretical average transaction rate of: 76.3 trx/s.

How to measure the average seek time?

For a disk with 8.5ms seek time and 4ms rotational latency, you can expect to spend about 12.5ms between the moment the disk `wants' to read your data and the moment when it actually starts reading it. This is the one number you are trying to make small. Thus, you're looking for a disk with low AST and high rotation (RPM) rates.

This value is in general not provide by you hardware vendor. You have to measure it yourself and prepare some tests for it.

From the theoretical calculation we received:

AAT = 13.1 ms, resp. 76.3 trx/s

A practical test with bonnie produced the following result:

```
# bonnie -v 3 -2000
...
               ---Sequential Output (nosync)--- ---Sequential Input-- --Rnd Seek-
               -Per Char- --Block--- -Rewrite-- -Per Char- --Block--- --04k (03)-
Machine    MB K/sec %CPU K/sec %CPU K/sec %CPU K/sec %CPU K/sec %CPU   /sec %CPU
master 3*2000 21749 92.7 41271 25.6 17523 16.6 18872 82.0 31925 16.4  102.1  1.2
```

This gives us 102.1 random seeks per second which accords to 102.1 trx/s or a AAT of 9.8 ms.

The output of:

```
mysql> SHOW ENGINE INNODB STATUS;
...
LOG
---
Log sequence number 0 11652658
Log flushed up to   0 11652391
Last checkpoint at  0 10815100
1 pending log writes, 1 pending chkp writes
65038 log i/o's done, 90.30 log i/o's/second
...
```

gives us a transaction rate of 90.3 trx/s or a AAT of 11.1 ms

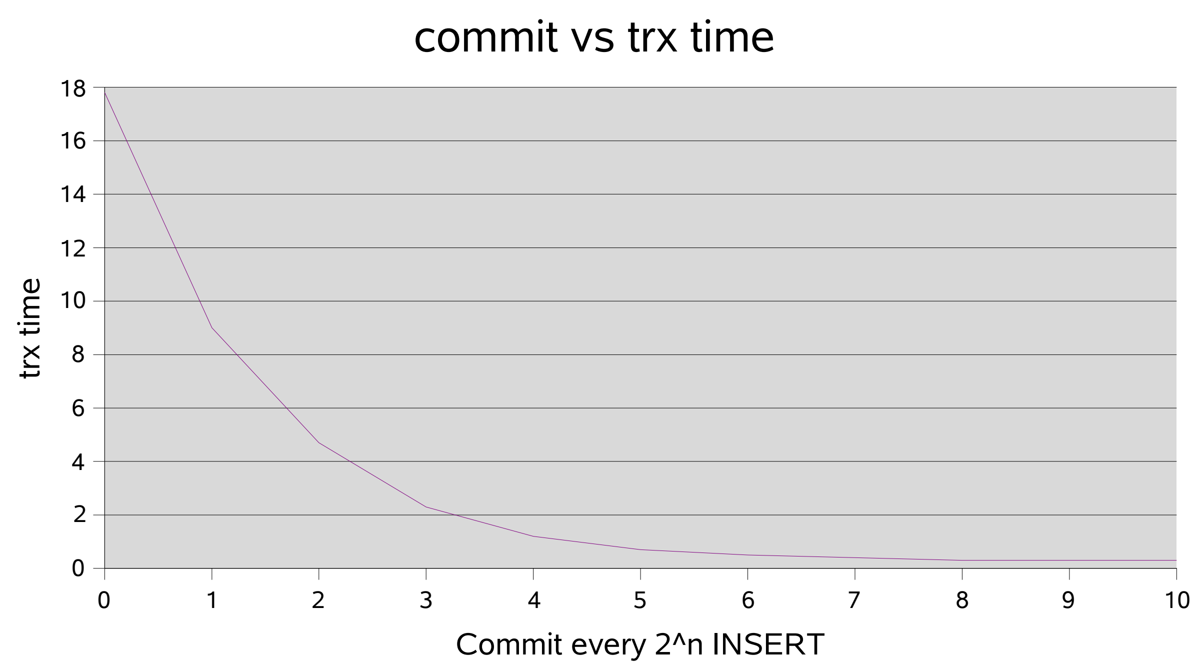
An other method is the tool writetest provided by [7]:

```
# ./writetest
Running: 1000 ops
Elapsed time: 0:0:8.819785
1000 operations:  113.38 operations per second
```

Which gives us a transaction rate of 113.4 trx/s or a AAT of 8.8 ms.

Or we measure our transaction throughput ourself with a test (see commit_demo.pl). This test took approx. 18 s for 1000 Inserts each follow by a COMMIT. This give approx. 55.6 trx/s or a AAT of 18.0 ms.

When we grouped more INSERTS into 1 transaction the throughput increased significantly:



Summary

	throughput	*AAT*	*Comment*
	[trx/s]	*[ms]*	*[-]*
Calculated value	76.3	13.1	
Bonnie	102.1	9.8	
InnoDB status	90.3	11.1	probably to low?
writetest	113.4	8.8	
commit_demo.pl	55.6	18.0	

How to increase transaction throughput

The following possibilities are available to increase transaction throughput:

1.) Improve transactions itself

* Make bigger transactions (less commits). See commit_demo.pl

2.) Improve performance of your IO system.
This goal can be achieved by more and faster disks as well as properly located data on the disk:
* Striping (RAID 1, 10 (and 5 for read only DB's)).
* Place hot areas on separate disks.
* Critical data to the outer cylinders (they are faster and hold more data).
* Place frequently used data near each other.

Literature

[1] http://duplex.hypermart.net/books/hards/029-031.html

[2] http://en.wikipedia.org/wiki/Seek_time

[3] http://en.wikipedia.org/wiki/Rotational_delay

[4] http://www.storagereview.com/guide2000/ref/hdd/perf/perf/spec/posSeek.html

[5] http://www.epinions.com/cmd-review-74AE-10919B0E-393D9E1E-prod1

[6] http://www.sleepycat.com/docs/ref/transapp/throughput.html

[7] http://www.sleepycat.com/docs/ref/transapp/writetest.cs

Tools

iozone

tiobench

bonnie / bonnie ++

hdparm

commit_demo.pl

```perl
use strict;
use warnings;

use DBI;

my $commit_every = 200;

# Start with: time perl commit_demo.pl

#CREATE TABLE test.commit_demo
#(
#    id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY
#) ENGINE=InnoDB
#;
#
#GRANT INSERT ON test.commit_demo
#   TO 'test'@'localhost' IDENTIFIED BY 'test'
#;
#
#DROP TABLE test.commit_demo;

my ($dbh, $sql, $sth);

$dbh = DBI->connect('DBI:mysql:test:localhost'
                  , 'test', 'test'
                  , { RaiseError => 1 }
                   );

$dbh->{AutoCommit} = 0;

$sql = "
  INSERT INTO test.commit_demo (id)
  VALUES (?)
";
```

```
$sth= $dbh->prepare($sql);

$| = 1;   # Write print output synchronous

my $i = 1;
while ( $i <= 1000 ) {

  $sth->execute('NULL');

  if ( $i % $commit_every == 0 ) {
    print ".";
    $dbh->commit();
  }

  $i++;
}
$dbh->commit();
$sth->finish;

$dbh->disconnect
```