Optimizing Database Read Performance

1. Database denormalization
2. Database storage engines (MyISAM vs InnoDB)
3. Distributed databases (partitioning, NoSQL, memcached)
4. Solid State Drive (SSD)

Database Normalization

Normalized form

- intended for online transaction processing (OLTP)
- reduce duplication of data
- simplify adding, updating, removing records

Denormalized form

- intended for online analytical processing (OLAP).
- optimized for read performance

Example: Download Genotype Data

- Using the current database schema requires a query that joins 5 tables. It takes several minutes to execute.

  Normalized schema

- To increase performance we reduce the number tables to 3 by adding columns from the linked tables onto the allele_cache table

  Denormalized schema

Performance improvement

	Read Query	Read Query with sort
Normalized (5 table join)	1.0 sec	6.0 sec [1]
Denormalized (3 table join)	0.5 sec	0.5 sec [2]

[1] SELECT lr.line_record_name, m.marker_name AS name, CONCAT(a.allele_1,a.allele_2) AS value FROM markers as m, line_records as lr, alleles as a, tht_base as tb, genotyping_data as gd WHERE a.genotyping_data_uid = gd.genotyping_data_uid AND m.marker_uid = gd.marker_uid AND tb.line_record_uid = lr.line_record_uid AND gd.tht_base_uid = tb.tht_base_uid AND tb.experiment_uid IN (100) ORDER BY lr.line_record_name, m.marker_uid;

[2] SELECT lr.line_record_name, m.marker_name AS name, alleles FROM allele_cache, markers as m, line_records as lr where m.marker_uid = allele_cache.marker_uid AND lr.line_record_uid = allele_cache.line_records_uid AND experiment_uid IN (100) ORDER by allele_cache.line_records_uid, allele_cache.marker_uid;

MySQL implementation of denormalized tables

- The denormalized tables are additional tables that are added to the schema. They are usually dropped and recreated once every day.
- The normalized tables are used for additions, updates and deletions
- In Oracle these tables are called *materialized views*. In MS SQL these are called *Indexed Views*

Database Storage Engines

- The MySQL database offers a choice of storage engines for specific uses
- InnoDB – default storage engine for the last 2 years. Has the best reliability and works well under high load.
- MyISAM – previously the default storage engine. Is more space efficient. Not faster, less reliable, and does not perform well under heavy loads

Distributed Databases: Partitioning

- Also know as sharding, splitting a table into groups of rows
- MySQL implementation - portions of the table are stored as separate tables in different locations, this is a standard MySQL plugin
- The user selects the partitioning function
- Advantages

  1. query optimization
  2. higher throughput
  3. allows data to fit on smaller hard drive or limited memory

Partitioning

- Example code to create a partition

  CREATE TABLE ti (id INT, amount DECIMAL(7,2), tr_date DATE)

  PARTITION BY HASH(MONTH(tr_date))

  PARTITIONS 6

- Example application

  A very large table of SNP data could be partitioned by experiment or year. Then a query by that parameter would only retrieve from one of the partitions. This could improve speed by roughly 10x

NoSQL

- Used to describe a lightweight database that does not use SQL queries
- Designed for high throughput
- MySQL implementation use memcache and InnoDB storage engine
- Another open source implementation is Apache Cassandra (developed by Facebook)

Memcached

- A general purpose distributed memory cache system
- Used by Facebook, YouTube, Twitter, Zillow, Yahoo, Amazon, etc
- The advantages are that it massively scales out, client does one layer of hashing to reduce load
- It is trivial to add more nodes to the system

Memcached example

- Convert database to use memcached is simple

```
function get_foo(int userid) {
result = db_select("SELECT * FROM
users WHERE userid = ?", userid);
return result; }
```

- Query database using memcached PHP

```
$memcache = new Memcache;
$memcache->connect('localhost', 11211) or die ("Could not connect");
echo “create object”
$tmp_object = new stdClass;
$tmp_object->str_attr = 'test';
$tmp_object->int_attr = 123;
echo “save in cache”
$memcache->set('key', $tmp_object, false, 10) or die ("Failed to save data at the server");
echo “get from cache”
$get_result = $memcache->get('key');
```

MySQL with Memcached

- MySQL 5.6 has memcahced built in
- Other version can be compiled and uses as an API
- Can use threaded operations which allow it to utilize many CPU on one system

Memcached performance

- In a one to one comparison it may not be faster than SQL but this is not the goal
- Memcached goal is scalability so works best under heavy load

Solid State Drive - SSD

- See MySQL 2010 presentation slide 22-29