CSCI317 Database Performance Tuning

Typical Database Performance Problems

Dr Janusz R. Getta

School of Computing and Information Technology -
University of Wollongong

Typical Database Performance Problems

Outline

What can go wrong ?

- Problems related to implementation of user applications
- Problems related to configuration of database server
- Problems related to transaction processing
- Problems related to database design
- Problems related to physical database design
- Problems related to operating system
- Problems related to hardware

Typical Database Performance Problems

Outline

What can go wrong ?

- Problems related to implementation of user applications
- Problems related to configuration of database server
- Problems related to transaction processing
- Problems related to database design
- Problems related to physical database design
- Problems related to operating system
- Problems related to hardware

Problems related to implementation of user applications

Inefficient SQL

Relational schemas

```
CUSTOMER(cnumber, name, address) PK = (cnumber)
ORDERS(onumber, cnumber, total) PK =(onumber), FK = (cnumber)
```

Find the total number of orders submitted by a customer whose number is 007

Inefficient SQL

```
SELECT COUNT(*)
FROM ORDERS JOIN CUSTOMER
            ON ORDERS.cnumber = CUSTOMER.cnumber
WHERE CUSTOMER.cnumber = 007;
```

Correct SQL

```
SELECT COUNT(*)
FROM ORDERS
WHERE ORDERS.cnumber = 007;
```

Problems related to implementation of user applications

Domination of procedural code over SQL

Find the names of customers who submitted orders worth more than 1000

Inefficient embedded SQL

```
for customer in CUSTOMER do
 for order in ORDERS do
  if customer.cnumber = order.cnumber and order.total > 1000 then
   display customer.name
```

Correct SQL

```
SELECT name
FROM ORDERS JOIN CUSTOMER
            ON ORDERS.cnumber = CUSTOMER.cnumber
WHERE total > 1000;
```

Problems related to implementation of user applications

Domination of SQL over procedural code

Find the orders with the second highest value of total

Inefficient SQL

```
SELECT *
FROM ORDERS
WHERE ORDERS.total = ( SELECT MAX(total)
                       FROM ORDERS
                       WHERE total <> ( SELECT MAX(total)
                                        FROM ORDERS ) );
```

A more efficient solution reads a relational table `ORDERS` row by row and keep the current largest value of total and the current second largest value of total, update both values when it is necessary

Typical Database Performance Problems

Outline

What can go wrong ?

- Problems related to implementation of user applications
- Problems related to configuration of database server
- Problems related to transaction processing
- Problems related to database design
- Problems related to physical database design
- Problems related to operating system
- Problems related to hardware

Problems related to configuration of database server

Incorrect configuration of DBMS, for example too small size of shared memory area, too small size of database cache, incorrect allocation of relational tables to database caches, etc

Incorrect configuration of persistent storage, for example, log files located on the same persistent storage device as database files, frequently used relational tables located on the same persistent and slow storage device, etc

Not enough database processes, e.g. one writer process in a heavy insert/update/delete database environment

Too small data block for large and frequently read relational tables

To small data transmission buffer

Typical Database Performance Problems

Outline

What can go wrong ?

- Problems related to implementation of user applications
- Problems related to configuration of database server
- Problems related to transaction processing
- Problems related to database design
- Problems related to physical database design
- Problems related to operating system
- Problems related to hardware

Problems related to transaction processing

Incorrect implementation of database transaction

Transaction T_i

Update an entire relational table

```
while not eot(ORDERS)
 read(ORDERS.total);
 write(ORDERS.total+1);
end
COMMIT;
```

Transaction T_k

Update a row in a relational table

```
read(ORDERS.total, onumber = 777)
write(ORDERS.total-20);
COMMIT;
```

Transaction T_i blocks transaction T_k because COMMIT of T_i is performed after all updates are committed

Problems related to transaction processing

Incorrect isolation level set for a database transaction

Too high isolation level

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

Update an entire relational table

```
UPDATE ORDERS
SET TOTAL = TOTAL + 0.1 * TOTAL;
COMMIT;
```

Appropriate isolation level

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
```

Update an entire relational table

```
UPDATE ORDERS
SET TOTAL = TOTAL + 0.1 * TOTAL;
COMMIT;
```

Typical Database Performance Problems

Outline

What can go wrong ?

- Problems related to implementation of user applications
- Problems related to configuration of database server
- Problems related to transaction processing
- Problems related to database design
- Problems related to physical database design
- Problems related to operating system
- Problems related to hardware

Problems related to database design

Logical database design not consistent with the database applications

2NF relational schema

```
R(driver, truck, capacity)
driver  → truck, truck → capacity
```

A sample decomposition into BCNF relational tables

```
D(driver, truck)
T(truck, capacity)
```

Find all drivers who use trucks that have a capacity greater than 100

Too complex SQL

```
SELECT driver
FROM D JOIN T
       ON D.truck = T.truck
WHERE capacity > 100;
```

Problems related to database design

Logical database design not consistent with the database applications

2NF relational schema

```
R(driver, truck, capacity)
driver  → truck, truck → capacity
```

Another decomposition into BCNF relational tables

```
D(driver, truck)
C(driver, capacity)
```

Find all drivers who use trucks that have a capacity greater than 100

Simpler SQL statement

```
SELECT driver
FROM C
WHERE capacity > 100;
```

Problems related to database design

Database physical design not consistent with database applications

A relational schema

```
ORDERS(onumber, cnumber, total) PK =(onumber), FK = (cnumber)
```

Find all orders whose total value equal to 1000

Sample SQL

```
SELECT *
FROM ORDERS
WHERE total = 1000;
```

Missing index on `ORDERS(total)` forces a full scan of a relational table `ORDERS`

Problems related to database design

Overnormalization

BCNF relational schemas

```
STREETS(street, zcode)    CITIES(city, zcode)
PK =(street, zcode)       PK = (zcode)
BCNF                      BCNF
```

Find all streets in Sydney

Complex SQL

```
SELECT street
FROM STREETS JOIN CITIES
             ON STREETS.zcode = CITIES.zcode
WHERE city = 'Sydney';
```

Problems related to database design

Denormalization

3NF and not BCNF

```
LOCATION(city,street,zcode)
PK =(city,street), CK = (zcode,street)
city,street  → zcode zcode  → city
```

Find all streets in Sydney

Simpler SQL

```sql
SELECT street
FROM LOCATION
WHERE city = 'Sydney';
```

Problems related to database design

Using surrogate keys

Artificial identifiers

```
BUILDING(bID, campus, bnumber, type) PK = (bID), CK = (campus, bnumber)
ROOM(bID, rnumber, area) PK =(bID, rnumber), FK = (bID)
```

Find all names of campuses and numbers of buildings that have at least one room whose area is greater than 100

Complex SQL

```
SELECT campus, bnumber
FROM BUILDING JOIN ROOM
              ON BUILDING.bID = ROOM.bID
WHERE area > 100;
```

Problems related to database design

Using surrogate keys

No artificial identifiers

```
BUILDING(campus, bnumber, type) PK = (campus, bnumber)
ROOM(campus, bnumber, rnumber, area) PK =(campus, bnumber, rnumber),
                                     FK = (campus, bnumber)
```

Find all names of campuses and numbers of buildings that have at least one room whose area is greater than 100

Simple SQL

```
SELECT campus, bnumber
FROM ROOM
WHERE area > 100;
```

Typical Database Performance Problems

Outline

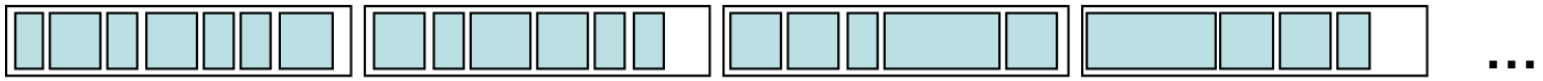
What can go wrong ?

- Problems related to implementation of user applications
- Problems related to configuration of database server
- Problems related to transaction processing
- Problems related to database design
- Problems related to physical database design
- Problems related to operating system
- Problems related to hardware

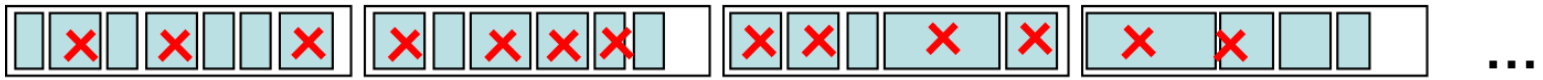
Problems related to physical database design

Fragmentation of persistent storage

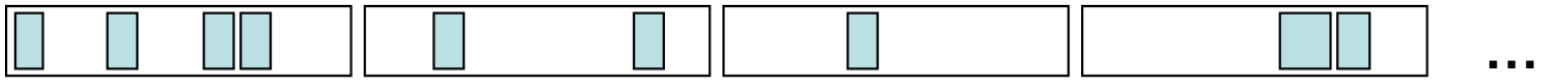
A relational table stored in the data blocks



Some rows are deleted



After deletions persistent storage is not automatically defragmented



Typical Database Performance Problems

Outline

What can go wrong ?

- Problems related to implementation of user applications
- Problems related to configuration of database server
- Problems related to transaction processing
- Problems related to database design
- Problems related to physical database design
- Problems related to operating system
- Problems related to hardware

Problems related to operating system

Incorrect file system used to store database files

Incorrect limits of batch jobs

Incorrect logging

Incorrect communications control

Incorrect priority assignments

and the others ...

Typical Database Performance Problems

Outline

What can go wrong ?

- Problems related to implementation of user applications
- Problems related to configuration of database server
- Problems related to transaction processing
- Problems related to database design
- Problems related to physical database design
- Problems related to operating system
- Problems related to hardware

Problems related to hardware

Running a complex database server on low performance hardware, for example, running Oracle on a laptop

References

D. Shasha and P.Bonnet Database Tuning Principles, Experiments, and Troubleshooting Techniques, Morgan Kaufmann, 2003, chapter 1