Database tuning

Viet-Trung Tran
SoICT

Outline

- Course organization
- Introduction to database tuning
- Basic principles of tuning

Online resource

- https://www.facebook.com/groups/trungtv.students/
    - **Discussion**
    - Exercise
    - Middle & final grade
    - Slides and everything else
- Email: trungtv@soict.hust.edu.vn

Course organization

- 11 weeks of lectures
- 4 weeks for team working quests
  - Team of 5
    - Challenge each others with real-work issues

Topics in this course

- Overview
- Query tuning
- Index tuning
- Concurrency tuning
- Modern database systems: NoSQL & Big data

What is database tuning?

- Activity to make database application running faster
  - Faster I/O operations: INSERT, SELECT, DELETE, UPDATE,
- And also optimizing storage space, network usage, etc.
- A 5% improvement is important

Tuning parameters

- Everything that make sense
  - Faster disk
  - More Ram
  - Effective index
  - Good queries
- There is always a cost/trade-off
  - Some time cost is low and the benefit very high

Tuning between theory and practice

- Practitioner
  - learning by experiences
  - Ex. Never use aggregate functions (AVG) when transaction response time is critical
  - Problem: AVG can be ok if less tuples
- Theoretician
  - Learning by mathematical models
  - Ex. Different between indexes
  - Problem: rely on ideal assumptions (rare in reality)

Database tuner

- Understand and apply **principles**
  - Understanding: the problem is not about AVG, but scanning large amount of data (which AVG often does…)
  - **Principle: Do not scan large amount of data in concurrency**
  - **Apply principle wisely**

Five basic tuning principles

- Think globally, fix locally
- Partitioning to break bottom necks
- Start-up costs are high, running costs are low
- Render on the server what is due on the server
- Be prepared for trade-offs

Think globally, fix locally

- Disk activity is high, what to do?
- Solution 1: buy more disk
- Solution 2: Speedup queries with longest runtime
- Solution 3: Speedup queries with largest share in runtime

Partitioning breaks bottom necks

- Rarely all parts of a system are saturated
- Partitioning strategies
  - Divide load over more resources (add lanes)
  - Spread road over time (avoid rush hours)

Start-up costs are high, running costs are low

- Reading operation
  - Disk seek is so expense
  - Continuous read is cheep
- Conclusion
  - Frequently scanned tables should be serialized sequentially on disk
  - Frequent query that projects few columns: vertically partition table – column based organization

Network latency

- Sending many small messages vs. sending little big message
- Ex. Sending 1 byte packet is almost as expense as sending 1 KB packet

Query overhead

- Query vs. Store procedure (compiled query)
- Compile often executed queries

Connection overhead from programing languages

- Open connection
  - – Significant overhead
  - – Establish connection
  - – User authentication
- Connection caching and pooling
- Do one SELECT and loops over results vs. Doing SELECT in the loop

- Lesson learned
  - Obtain results with the fewest possible startups

Be prepared for trade-offs

- Making one query faster may slow down other queries
- Index can make certain queries faster, but
  - Addition disk space required
  - Slow down insert, update