Performance Tuning

Oracle Tutorials

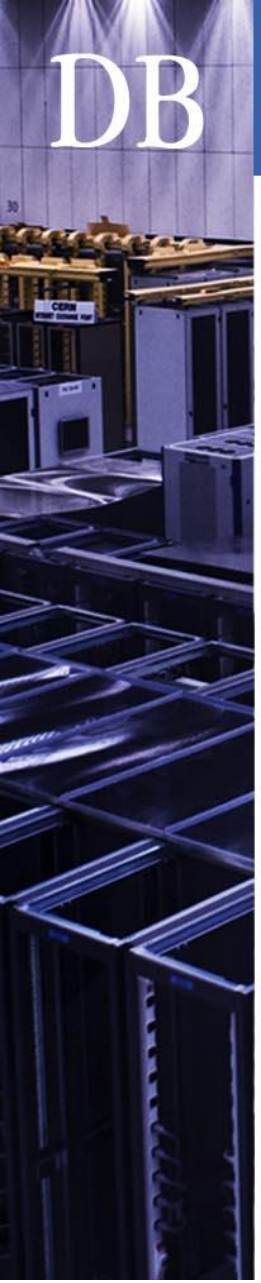
CERN

June 8th, 2012

CERN IT Department
CH-1211 Geneva 23
Switzerland
www.cern.ch/it

Outline

- Introduction – Luca Canali
- Application instrumentation – Chris Roderick
- Tools – Dawid Wojcik
- Demo – Eric Grancher



Why Performance Tuning?

- Need faster interaction between application and database
  - Help DB engine to better serve your needs

- Why is it relevant and interesting?
  - Performance can be highly visible in critical projects
  - It's about resources usage optimisation
  - Performance tuning requires logical thinking, systematic approach and creativity

CERN IT Department
CH-1211 Geneva 23
Switzerland
www.cern.ch/it

Database SERVICES

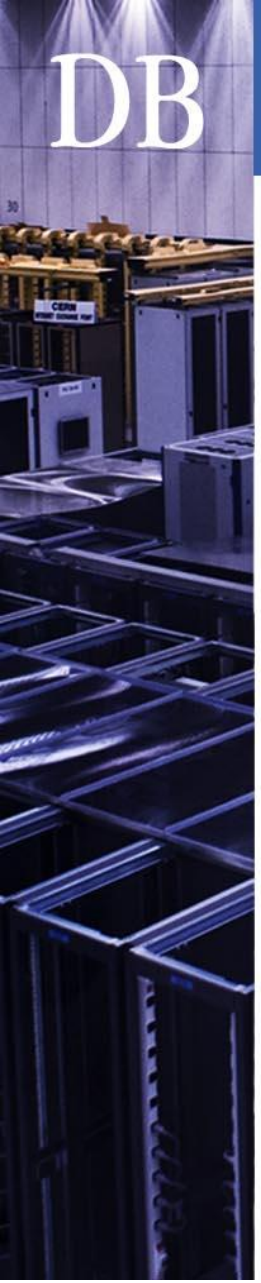
CERN

Examples

- Examples of cases that stress database engine performance:
  - High rate of concurrent transactions
  - High rate of queries
  - Queries that scan large amounts of data
  - Complex joins and aggregations on data
  - Strict requirements on response time

Proactive Tuning - Design

- Design phase
  - Foundations for future performance
- Example
  - Define logical and physical structure of the table
  - Define which tables to use, which indexes, etc
- Typical mistake
  - requirements change after design and we are stuck with suboptimal DB structure
- Critical:
  - Involve application owner (get the use case right)
  - involve DB experts



Guidelines

- Development Lifecycle
  1. Development DB
  2. Pre-production tests (integration DB)
  3. Stress testing (test DB)
  4. Production (reactive tuning)

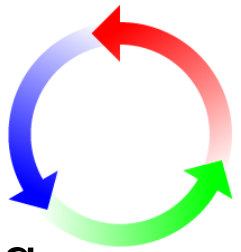
- In other words:
  - We promote extensive tests
  - Tests done in a production-like environment
    - Test with realistic size of data
    - Test with concurrent users

Guidelines


Development service
Validation service
Production service

Systematic Approach

- Tuning by understanding
- Example:
  - Have a clear view of what is the final goal
    - A good goal is something that matters to the end-users
  - Define the scope of the problem
    - Drilldown to the problem-area
  - Build a mental model of what is happening
    - Which parts are involved? How do they interact?
  - Perform tests and relevant measurements
    - Where is time spent? Are there bottlenecks?
    - Confirm or disprove the model, check the goal
  - Possibly propose change(s) + further testing



Oracle is Instrumented

- Oracle database access is heavily instrumented
  - It is not a black box!

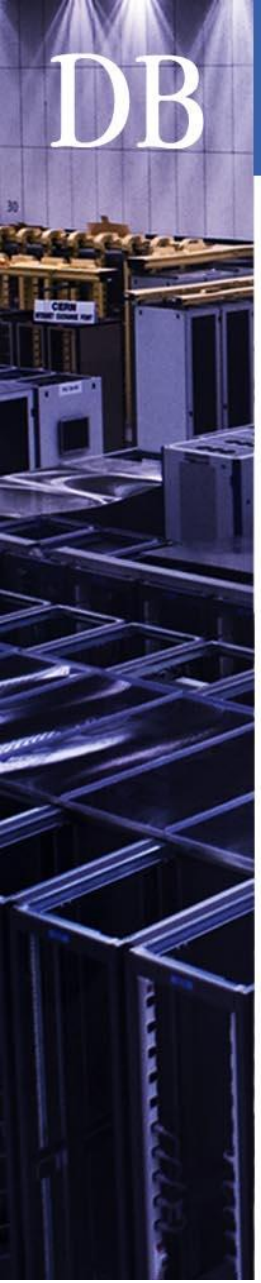
- Basic idea:
  - It is possible to know what Oracle is doing at a given time
- Example:
  - It is possible to measure where time is spent
    - On which SQL statement
    - On what part of the SQL execution
    - How much time is spent on CPU
    - How much time is spent waiting for disk access, etc

Profiling Execution

- What we want to achieve
  - Find out where time is spent during execution
- Drill down on the relevant code part
  - Profile your code in the relevant parts
    - Some times this is slower than actual Oracle SQL time
  - Profile Oracle execution
    - Time-based details of SQL, CPU usage, I/O, etc
- Tools and techniques
  - Details in the following presentations
  - This includes, execution plan, SQL traces, OEM

Finding Bottlenecks

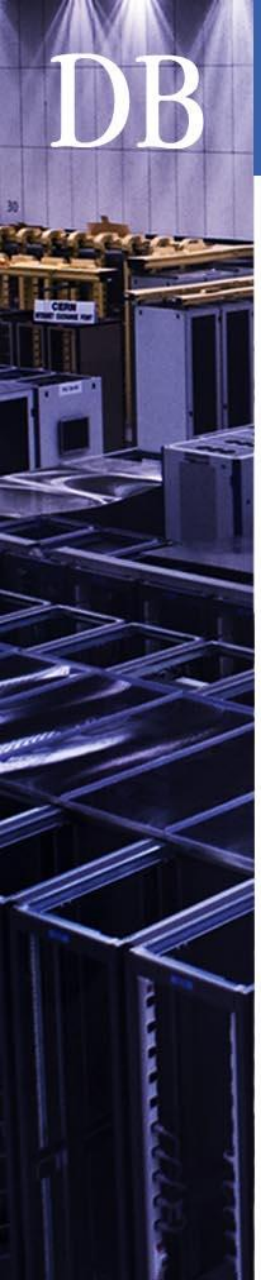
- There is always a limiting factor
  - Measurements can show if and where it is on DB access
- Detail time spent by DB processing to find bottlenecks
- Examples:
  - IO read time (throughput)
  - IO random read time (latency)
  - Lock contention (transactional wait)
  - CPU time for SQL execution



Lateral Thinking

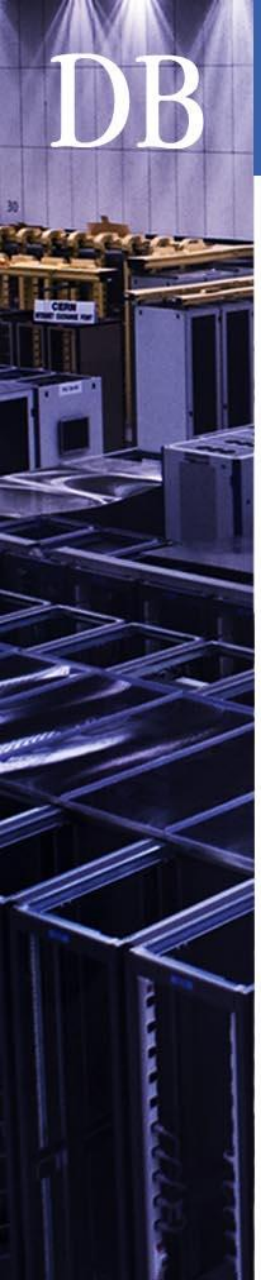
- A different path to the same goal
  - Turn the problem around as opposed to tune the sub-steps one by one

- How to make it work. Example:
  - Have a clear understanding of the use case
  - Build a model of what is happening in the application and in Oracle
    - i.e. understand why the implementation doesn't perform
  - Know enough of the application and Oracle to get to the goal via a different path
    - Know who to involve/ask for help



What you Need to Know

- Know your application usage of Oracle
- Oracle knowledge
  - Know DB/Oracle features that can help
    - Indexes, partitioning, etc
  - Know basic Oracle internals/architecture
- Know where to look for documentation
  - http://oracle-documentation.web.cern.ch
- Know the tools available to gather data
  - See following presentations on tools
- Know how to ask for help
  - Provide precise problem info to database experts



Key Points of the Introduction

- Performance is very important
- Tuning the design is most effective
- Follow a development lifecycle process
  - development -> test-> (stress test) -> production
- Analyse perf issues by understanding
  - Clear goal expressed with user-centric metrics
  - Gather relevant data and build model
- Use application and SQL profiling
  - Understand where time is spent
  - Understand where bottlenecks are

CERN IT Department
CH-1211 Geneva 23
Switzerland
www.cern.ch/it