

# Architecture for System Performance & Scalability

Qinghui Zeng  
Chief Performance Architect  
Oracle

March 2009

## Agenda

- ✓ What are the System Performance & Scalability
- ✓ How to Design & Implement a Performing & Scalable System
- ✓ Hardware and Performance
- ✓ Methodologies in Performance & Scalability Tests
- ✓ Common Techniques, Tools & Utilities in Performance & Scalability Tests
- ✓ Q&A

## What are Performance & Scalability

- ✓ The Same as the Olympic Motto:
  - ✓ Swifter – faster online response time, faster batch & interface jobs
  - ✓ Higher – Support higher number of users & data volume
  - ✓ Stronger – more stable & robust system

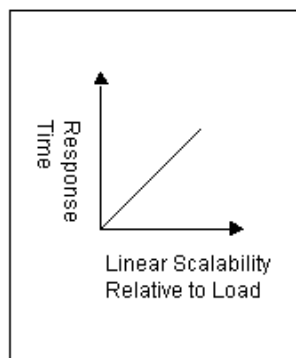
## Performance Principles

- ✓ Online Response Time:
  - ✓ Not to slow down users' productivity i.e.  $\leq 2$  seconds for a transaction
  - ✓ If not possible, not to impact the users' morale or make the users not come back
- ✓ Batch & Interface Jobs:
  - ✓ Meet the SLA
- ✓ System Resources :
  - ✓ Efficient on system resources to minimize hardware and platform software costs
- ✓ Optimal Investment:
  - ✓ Stop Tuning when meet the performance requirements

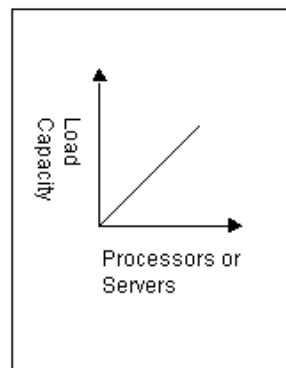
# Scalability Principles

- ✓ **Free from Structural Limitations**
  - ✓ No architectural & design issues to scale up and down
- ✓ **Linear Performance**
  - ✓ The hardware capacity requirements of the system should be directly proportional to the increased demand of the application (users, transactions, queries, etc.)
  - ✓ With the same hardware, the processing time should be directly proportional to the data volume to be processed
- ✓ **Scale Vertically & Horizontally**
- ✓ **Address Business Demands**
  - ✓ Scalability > the business demand

# Scalability Charts

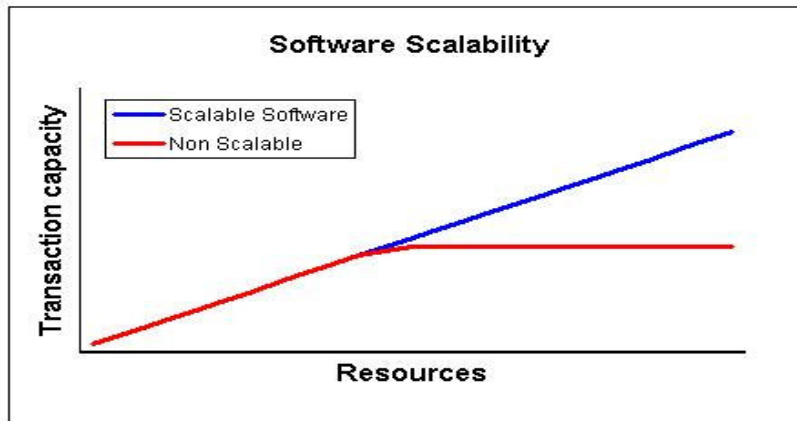


Linear Scalability Relative to Load



Linear Scalability Relative to System Resources  
(e.g. hardware)

## A System with Bottleneck



## Why Non-Performing or Non-Scalable

- ✓ Inefficient Application
- ✓ Unable to Use All System Resources Available Eg. Single Threaded Batch Job
- ✓ Resources Contention/Bottleneck
- ✓ Database Lock Contention
- ✓ Exceed the Platform Limitations - HW, O/S, Application Server, Database Server, Programming Language ...
- ✓ Unrealistic Business Requirements

## How to Get a Performing & Scalable System

1. Design the System with Performance & Scalability Considerations
2. Implement the System with Performance & Scalability Considerations
3. Test the System's Performance & Scalability
4. Tune the System for Performance & Scalability

## How to Get a Performing & Scalable System

1. Choose the Right Technologies for the Right Jobs
2. Minimize Workload (use the minimum amount of System Resources to do the same job) end to end
3. Parallel Processing
4. Balance Workload Across Threads, Instances and/or Nodes
5. Avoid Resources & Lock Contention
6. Hardware Capacity Planning and Choose the Right Hardware for the Right Jobs
7. Address the Bottlenecks

## Choose the Right Technologies for the Right Jobs

- ✓ Differences with Software Performance
  - ✓ A Test Case Result (1 million records from a certain table):
    - Insert Into a database table: 60k records/second
    - Building PL/SQL Array: 8k records/second
    - Create Java Objects: 2.5k records/second
    - 1:7:24 ratio on the same hardware
    - PL/SQL, Java and C/C++ need several times more memory than database table in most cases
  - ✓ Also the max memory structure allowed
- ✓ Differences between Software Versions
  - ✓ Oracle 10g is about 30% to 35% faster than 9i on PL/SQL Operations. Some times it is up to 3.4 Times faster with certain operations

## Programming Language and Performance

1. Do it in SQL
  2. If not SQL, a little PL/SQL
  3. If not PL/SQL, a little Java
  4. If not Java, a little C/C++
  5. If not C/C++, then think is it really necessary
- Tom Kyte

(For Data Centric applications, for example, more than 1,000 records in one transaction)

## Choose the Right Technologies for the Right Jobs

- ✓ Need Quantified Figures of Performance and Scalability from Proof of Concept Tests
  - ✓ For application and individual functionality
- ✓ Different Criteria for Data Centric VS Non-data Centric Transactions
- ✓ Optimize All the Test Cases in Their Ways
- ✓ Test Case Coverage:
  - ✓ Most Complex Transactions – If the Technology can do the job for us?
  - ✓ Top Transactions by total resources usage – Can the technology do the job efficiently?
  - ✓ Largest Transactions – If the Technology can scale to that volume?
  - ✓ Need to simulate your production cases

## Where to Improve Performance & Scalability

### By the Order of Optimize Sequence

1. Business Requirements
2. Database Logical Design
3. Database Physical Design
4. Application Design, Implementation & Database Indexing
5. Database Parameters
6. Database Maintenance
7. Disk I/O
8. O/S and Server
9. \*Business Requirements

## Where to Improve Performance & Scalability

### By the Order of Potential Improvements

1. Application Design, Implementation & Database Indexing
2. Database Physical Design
3. Database Logical Design
4. Database Parameters
5. Database Maintenance
6. Disk I/O
7. O/S and Server
8. \*Business Requirements

## How to Improve Performance & Scalability

- ✓ Efficiency is the Premise for Performance
  - ✓ Design your system for multi-threading and large number of users, but more important – make it efficient on single thread and single user first
- ✓ Performance is the Premise for Scalability
  - ✓ Design your system to scale vertically and horizontally, but more important – make it efficient on single thread and single user first
- ✓ How fast is called efficient (net CPU time)
  - ✓ By Database: 1,000 - 100,000 records/sec or 10,000 to 150,000 blocks/sec
  - ✓ By Java : 100 records/sec
  - ✓ Physical I/O: 5 – 10 ms/IO
- ✓ Too Many to Talk in One Session
  - ✓ Can Schedule Another Session Dedicated for This
- ✓ Just Mention a Few Key Points Here for Data Intensive Applications

## In General

- Choose the Right Technologies for the Right Jobs – Some Program Languages are much more efficient to handle large amount of data. Process large amount of data inside database
- Application Design, Implementation & Database Indexing are where you can get the most performance improvements
- Multi-threading
- Some Industrial Common Practices are NOT good for Performance & Scalability
  - Keep all business logics in Middle Tier
  - Database Independence
  - Code reusability & modularization for unsuitable scenarios
  - Driving Cursor at the lowest level or loop within loop
  - Loop to call Functions/Procedures VS Procedure call loops

## Database Applications

- Execution Plan is the key for SQL performance – SQL tuning is to influence SQL Execution Plan basically
  - Correct the database stats
  - Re-write the SQLs, even driving cursors
  - SQL hints
  - Review indexing
  - Review partitioning strategy and much more ...
- Access minimum amount of blocks and access the blocks in efficient ways
- Partitioning strategy to match the business data access pattern
- Driving cursor to match the partitioning strategy
- Optimal processing unit size (usually 100K to 1 million rows\*) - bulk processing vs. chunking
- Use Bind Variables

## Database Logical Design

- Normalization vs. Denormalization
  - The size of the N in the 1 – N Relationship
  - The fatness of the Parent and Child tables
  - Data Access Pattern: Parent only or Parent + Child join
- Consider to make them in one table when
  - N is small
  - Parent table is slim and tall
  - And Parent + Child often join together

## Database Physical Design

- There is a way to access data faster than by ROWID – sequential I/O vs. random I/O
- Clustering data to match the data access pattern can improve I/O efficiency by orders of magnitudes
  - Partitioning
  - Get all columns from self contained indexes
  - Index Organized Table
  - Sort data before store
  - Working tables
- Partitioning has great benefit to purge data as well
- Choose index type local/global/partitioned option carefully
- Set Intrans and Freelists\* equal to the number of CPUs

## Database Parameters Tuning

- System Global Area (SGA) Tuning
- Program Global Area (PGA) Tuning
- Optimizer is very sensitive to some parameters
  - optimizer\_mode
  - optimizer\_index\_caching
  - optimizer\_index\_cost\_adj
  - optimizer\_dynamic\_sampling
  - optimizer\_features\_enable
  - optimizer\_secure\_view\_merging
  - db\_file\_multiblock\_read\_count
  - ...

## Database Maintenance

- Cost Base Optimizer relies on database stats heavily. Keep the database stats accurate – handle the stats on the temporary tables smartly
- Keep the volatile tables and their indexes healthy – truncate or delete the left over data, shrink the table's High Water Mark and rebuild indexes after many or large deletions
- Purge and archive Data

## Java Applications

- Minimize object creation
- Minimize object copying
- Minimize object size
- If need to process a lot of data, let the database do it
- Balance the use of caching
- JDBC Driver version – usually not to let it lower than the database version!
- It is the same for other Programming Languages
  - C/C++: Compiler vendor, version and optimization level

## Online Applications

- Paging if it has more than 50 records
- Sorting Techniques
- Keep the CPU less than 80% busy and disk less than 60% busy
- For applications over WAN (Wide Area Network)
  - Minimize payload and network roundtrips
  - Both network bandwidth and latency are important
  - Application is too chatty –Remote Desktop may be the solution
    - Graphics Device Interface GUI (Windows Apps & HTML) vs. non-GDI GUI (images, Java Applet, Oracle Forms etc.)
    - A cursor blinking may generate 1 M Bit/Sec network traffic with non-GDI GUI

## Batch Programs

- Driving Cursor and Looping
- Bulk Processing or Chucking
- Multi-threading and Threading Design
- Balance the workload – dynamic workload allocation and do the largest work first
- Batch schedule optimization
- Fully utilize the system resources

## Analytical Applications

- Algorithm and the implementation are the keys
- Some job is simple for the developer, but expensive for the system to do them. e.g. Count, Sum, Min, Max, Average
- Incremental calculations
- Many ways to do the same job, but some ways are much more efficient than others
- The Magic of Mathematics

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2}. \quad \Rightarrow \quad \sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N x_i^2 - \bar{x}^2}.$$

## Hardware Performance Trend

- Much faster and/or more CPU cores, larger memory, faster sequential I/O speed, actually less total random I/O bandwidth
- Each hard drive's capacity was <1GB, is >100GB
- Each hard drive's IOPS has not changed much
- I/O bandwidth was >100,000 IOPS/TB and is <1,000 IOPS/TB now
- For Data Intensive Application, > 90% of time is on I/O -- I/O efficiency become more & more important for application scalability

## How Fast Can A Rotational HD Do?

- Use a 15K RPM Drive for Example
  - 1 rotation:  $60,000 \text{ ms} / 15,000 \text{ RPM} = 4 \text{ ms}$
  - Disk head to move 4 cm and stop in 4 ms: 1,000 G
  - Service Time = Wait Time + Disk Head Moving Time + Rotation Time + Read/Write Time + Transfer Time  $\sim 5 \text{ to } 10 \text{ ms}$
  - If Transfer Rate is 200 MB/Sec, it can transfer 1MB data in 5 ms – So Stripe size should be larger than 1 MB now

## Use Index or Full Table Scan?

- Sequential I/O in 5 ms
  - 1,024KB / 8 KB = 125 Blocks
  - 125 Blocks \* 50 records/Block = 6,250 records
- Random I/O in 5 ms
  - 1 Block, 1 to 50 records depending on application
- Threshold for FTS: was >10%, now is < 1%
- Effect of Clustering Data
  - Use online banking, cell phone statement and credit report for example, 1 transaction need to read 10's to 100's records belong to the same users. If we can store them in the same and continuous block(s), it will be only 1 I/O.

## A New Hardware Technology

- A New Revolutionary Storage Technology - Solid State Drive
  - Built with flash memory
  - No moving part
    - More reliable – Military and Aerospace grade
    - No seek time, 30-100  $\mu$ sec Access Time, up to 55,000 IOPS
    - Low power consumption and cooling cost
  - Improved Write Endurance from 1,000's to 100,000's writes
  - Much Faster Random Reads
  - Comparable Sequential Read and Write Speed
  - Need to Improve Random Write Speed
    - Large Cache
    - SANDISK's ExtremeFFS
    - ...

# Hardware Capacity Sizing

- Use one industrial standard as the measurement unit
  - CPU speed is very different from one to another among CPU architecture, generation and clock speed
  - TPC or Spec
- Too many factors to consider? - identify the driving factors
  - User driven or transaction volume driven?
- Use a benchmark or production system as the reference point
  - Add 20% as the contingency for benchmark result
- Batch driven or online driven – choose the higher one, cpu < 80% with online system
- Give it adequate memory - never let the system swapping
  - user driven or thread/CPU driven
- Size the disk for both space and I/O capacity
  - Spec is 100 to 200 IOPS per drive, only count 30 to 50 IOPS!
  - RAID level: Stripe and Mirror Everything (SAME)
  - Database sizing: focus on top 20 tables

# Performance Test Methodologies

- Atomic Testing & Tuning
  - Single user and single thread tests and Tuning
- Test-to-Scale
  - Test scalability of application and overall solution
  - Validates that the architecture is free of structural limits to scale-up
  - Full scale hardware not required for Test-to-Scale
- Benchmark
  - Performance tests that simulate a particular workload scenario to pass specific business metrics
  - Simulation of 'a day in the life' to understand peak capacity
  - Perform capacity planning for specific retailers
- Note: Always use Production like & volume data

## Stats to be Captured

- CPU Utilization (Sys, User & IOWait)
- I/O Utilization (IOPS and MB/Sec)
- Network Utilization
- Memory Utilization
- If it is Oracle Database
  - Database Stats: Statspack Report or Automatic Workload Repository (AWR)
  - SQL Trace
  - Set timing on

## Performance Test Tools

- Online Application Testing
  - Mercury (HP) LoadRunner and WinRunner
  - WAN Simulator
  - Stop Watch
- Batch
  - Any schedulers or manual kick off
  - Shell/batch scripts
  - The Unix time command
  - Timing logs

## Performance Tuning Tools

- Database (Use Oracle for Example)
  - \*\*SQL Trace + tkprof
  - \*\*Statspack/AWR
  - \*OEM Grid Control
  - Auto Trace
  - Explain Plan
  - PL/SQL Profiler
  - RDA (Oracle)
  - Your own scripts
- Java Application
  - JProbe
  - OptimizeIT
  - JProfiler

## Performance Monitoring Tools & Utilities

- System
  - \*\*Nmon and nmon Analyser (Interactive and tracing mode) (Aix & Linux)
  - \*Glance (Interactive and tracing mode) & gpm (HP UX)
  - \*Sar (all Unix)
  - \*PerfMon (Interactive and tracing mode) (Windows)
  - \*Iostat (all Unix)
  - \*Netstat (all Unix)
  - \*Top (all Unix)
  - Vmstat (all Unix)
  - Mpstat (Solaris)
  - Ps (all Unix)
  - Time

## How to Become an Excellent Perf Specialist

- ✓ Deep understanding on the business logic
- ✓ Deep understanding on how the application, database, system, network and storage work
- ✓ Skillful on using the diagnosis and monitoring tools
- ✓ Clear understanding on the diagnosis and monitoring reports and keen on the info on the report
- ✓ Know which part is the biggest problem
- ✓ Identify and address the root problems rather than the symptoms
- ✓ Understand the impact of the changes
- ✓ Keep learning on the new features
- ✓ Do you own research/proof of concept

## Question?

- Q & A