TISYA

Experience the Difference

TROUBLESHOOTING AND TUNING ORACLE DATABASE

© 2018 TISYA Software Solutions Pvt. Ltd.

Introduction

- Where do you Work
- What is your primary expertise and experience
- Why this Class

Objectives

- In this course you will learn:
  - How database uses Computer Resources
  - Importance of design for Performance
  - Relationship between Hardware and Performance
  - Identifying Performance Metrics using Wait Events and Ratio Analysis
  - Physical Design
  - Optimizing Storage Configuration
  - How to Read an AWR Report
  - Use the Oracle Server memory most effectively
  - Identify Bad SQL

Pre-requisites

- Good Understanding of Oracle Database Architecture
- Experience in Performance Troubleshooting and Tuning
- Suggested Pre-requisite - Oracle Database Performance Tuning Class

We will Think out of the Box

Case Study - 1

Application User Experience

Complaint – From Users

Symptoms Observed – Application Responding Very Slow

Observation – CPU utilization on Database Server – 10%

What is your Initial Response as a DBA

Case Study - 2

System Administrator Experience

Complaint – From SysAdmin

Symptoms Observed – CPU utilization @ 80% on DB Server

Observation – Increased load on Database

What is your Initial Response as a DBA

Case Study - 3

What to put in the Bowl?

Complaint – Not able to keep all items

Symptoms Observed – Bowl is Full

Observation – ???

Case Study - 3

What to put in the Bowl?

- **Can we optimize this?**

- **Do we need a larger Bowl? A.k.a – Increase Hardware**

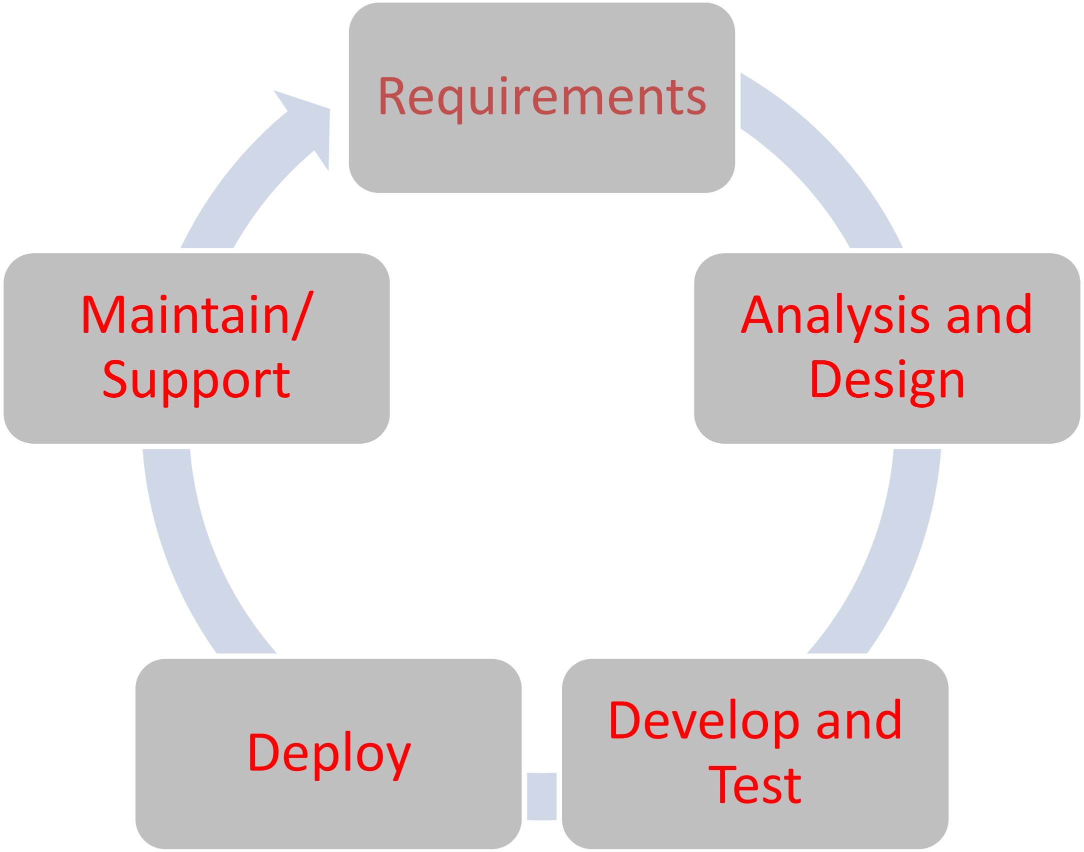
Do you have experience with

- Systems performing really Bad
- Systems performing really Well

- Was it by Chance? Was it a Surprise?
- Or was it Planned? Good or Bad Performance

What makes Software to Perform Well

- It depends on what we do at the Time of Deployment?
- Or does it takes all phases of Software Lifecycle



Understanding Database Execution and Resource Utilization

Chapter 1

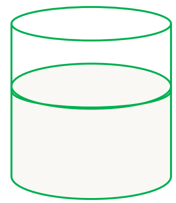
Agenda

- How does Oracle Database use its Resources
- Can we optimize Memory , CPU and I/O utilization

Critical Computing Components

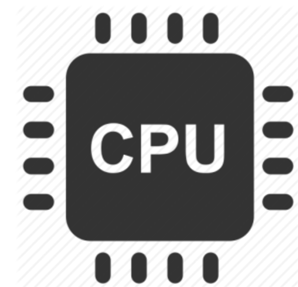
What are the main components? How are they utilized?

MEMORY

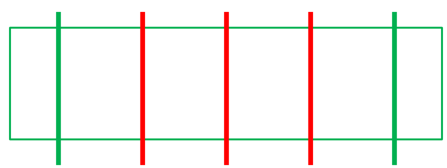


Full / Empty

CPU



Busy



Idle

HARD DISK

Critical Computing Components

TISYA

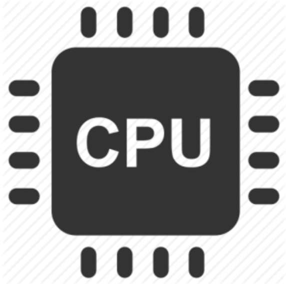
How are they abused?

MEMORY

Out of Memory

Out of Process Memory

Memory leak



CPU

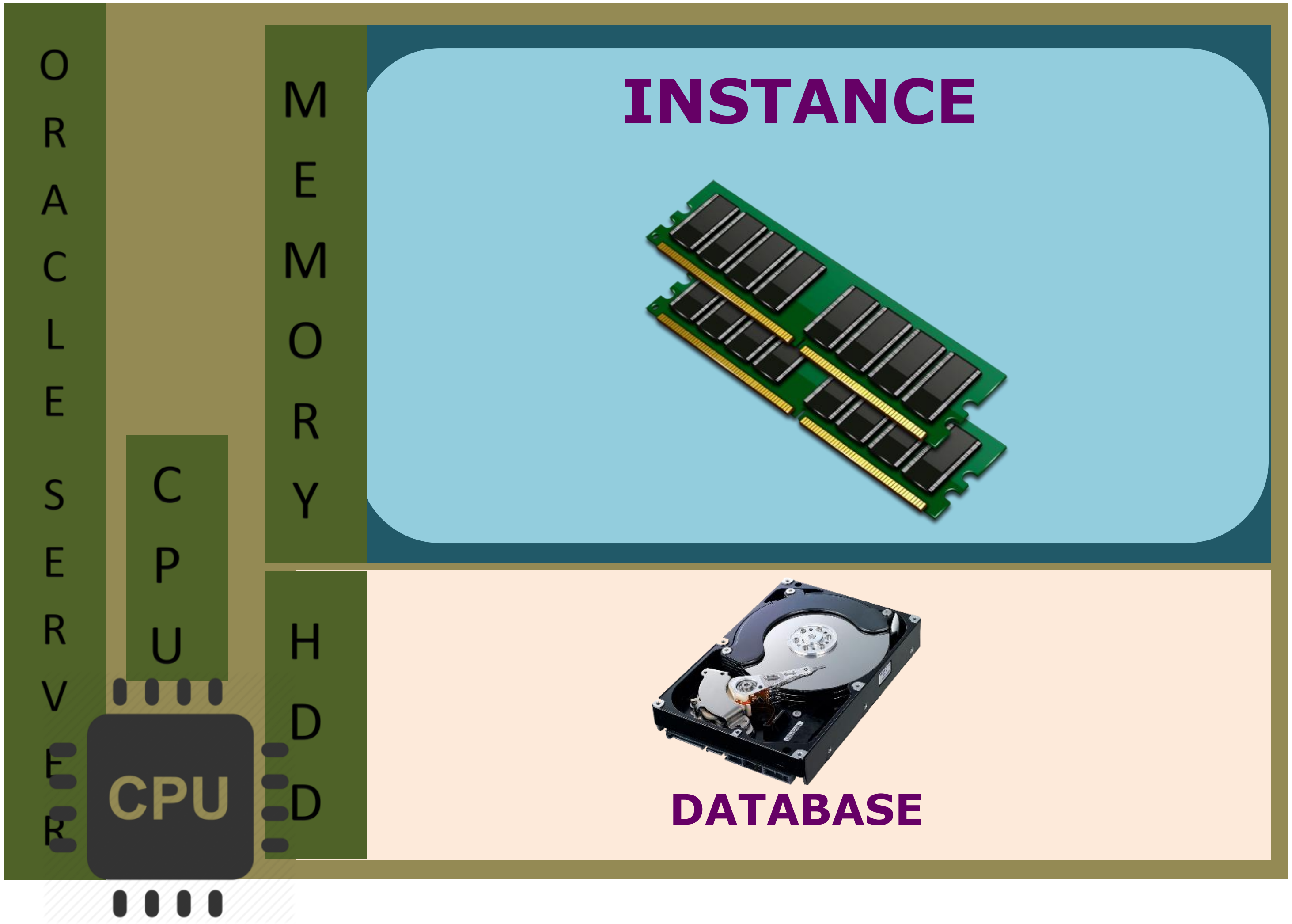
CPU 100% usage

HARD DISK

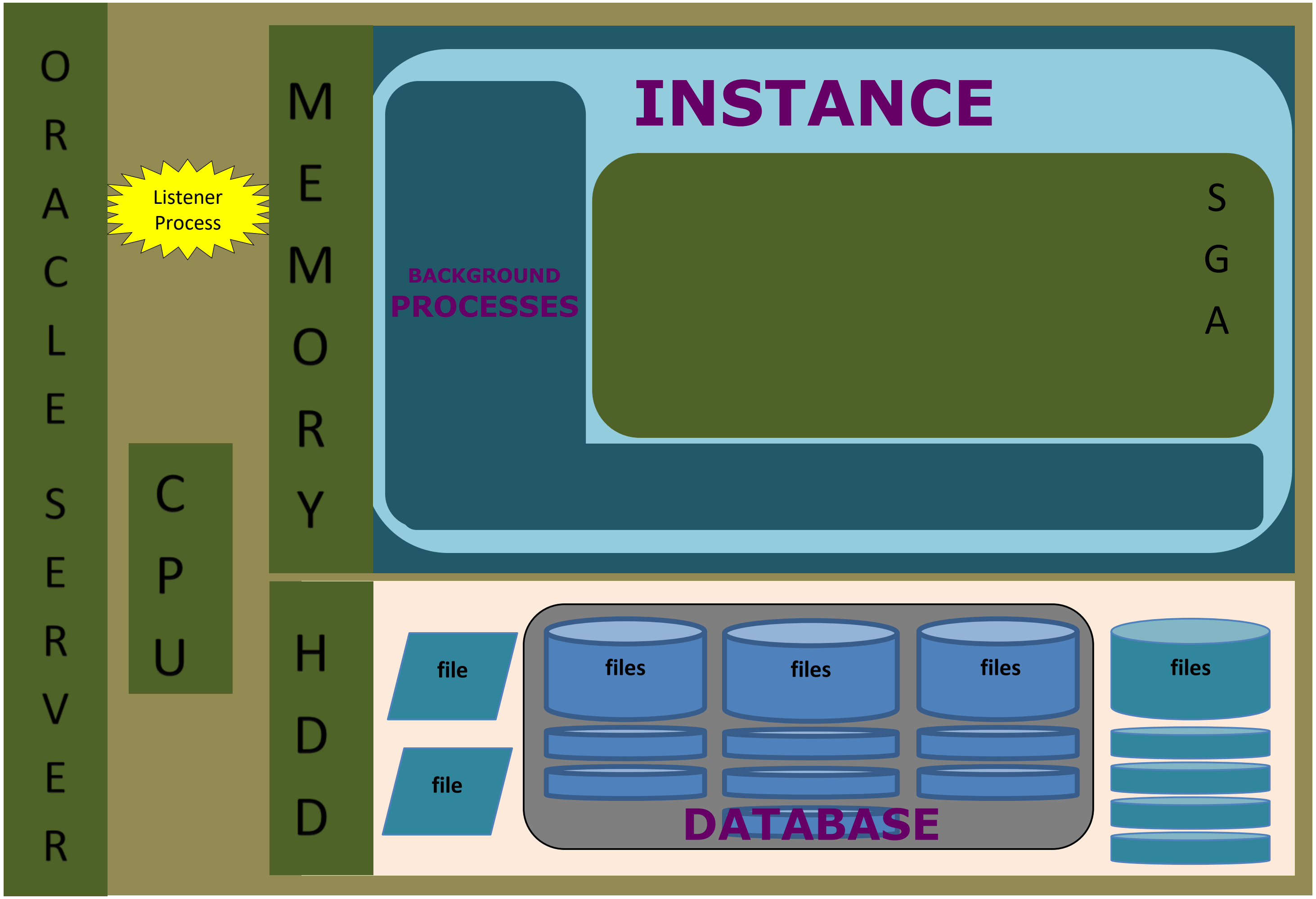
Hard disk full

Excessive I/O

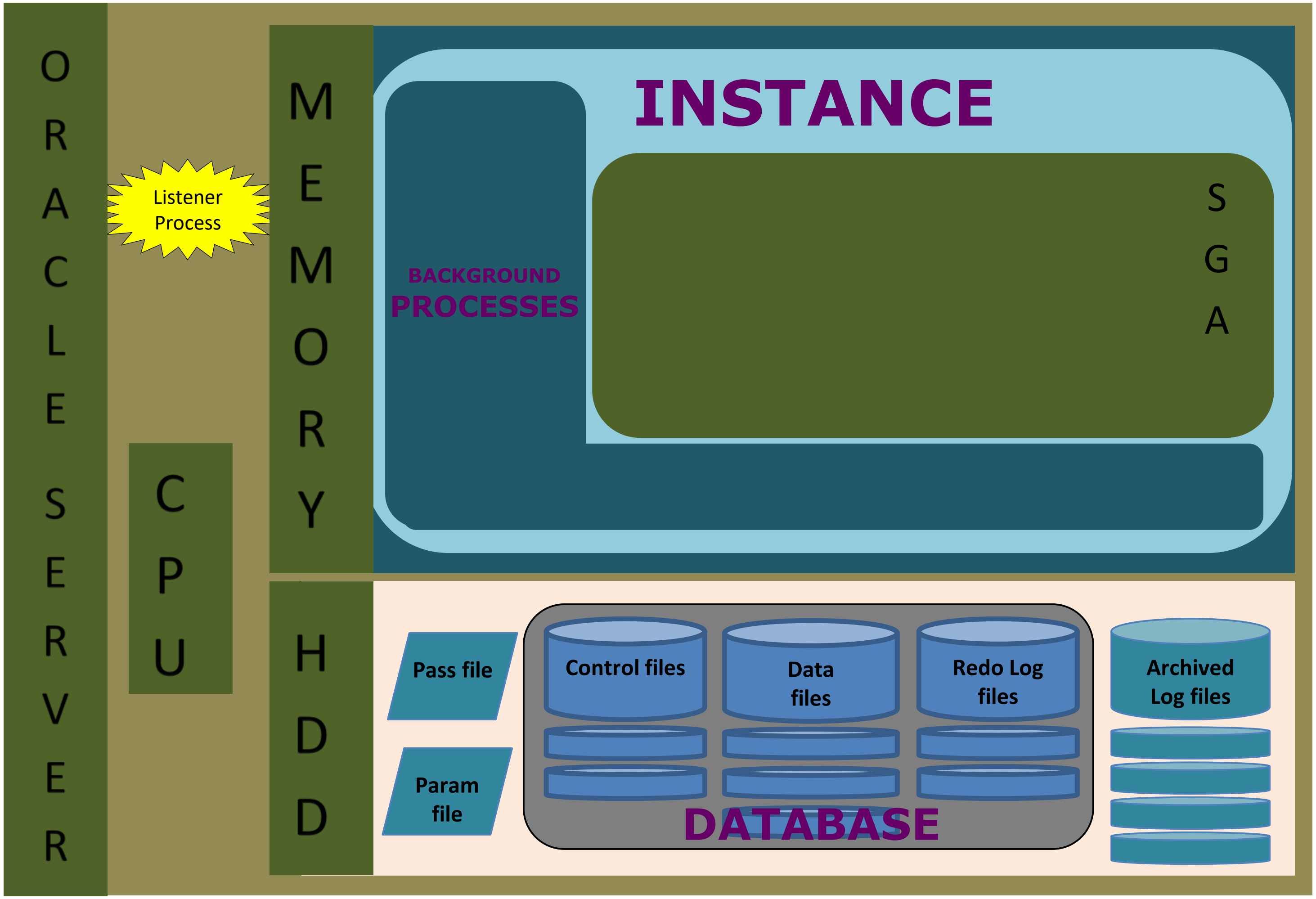
Oracle Computing Components



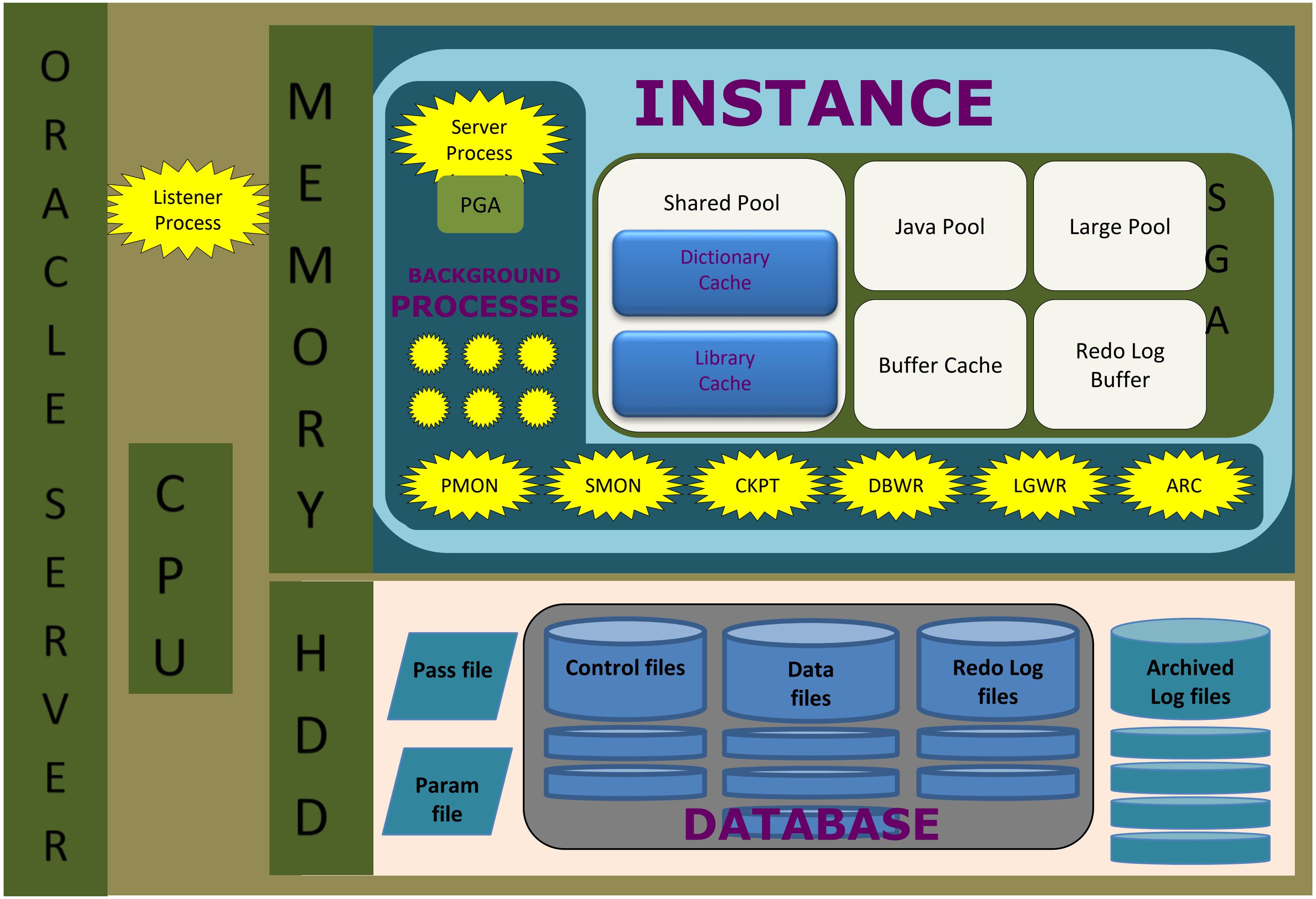
Oracle Architecture



Oracle Architecture - Database



Oracle Architecture - Instance

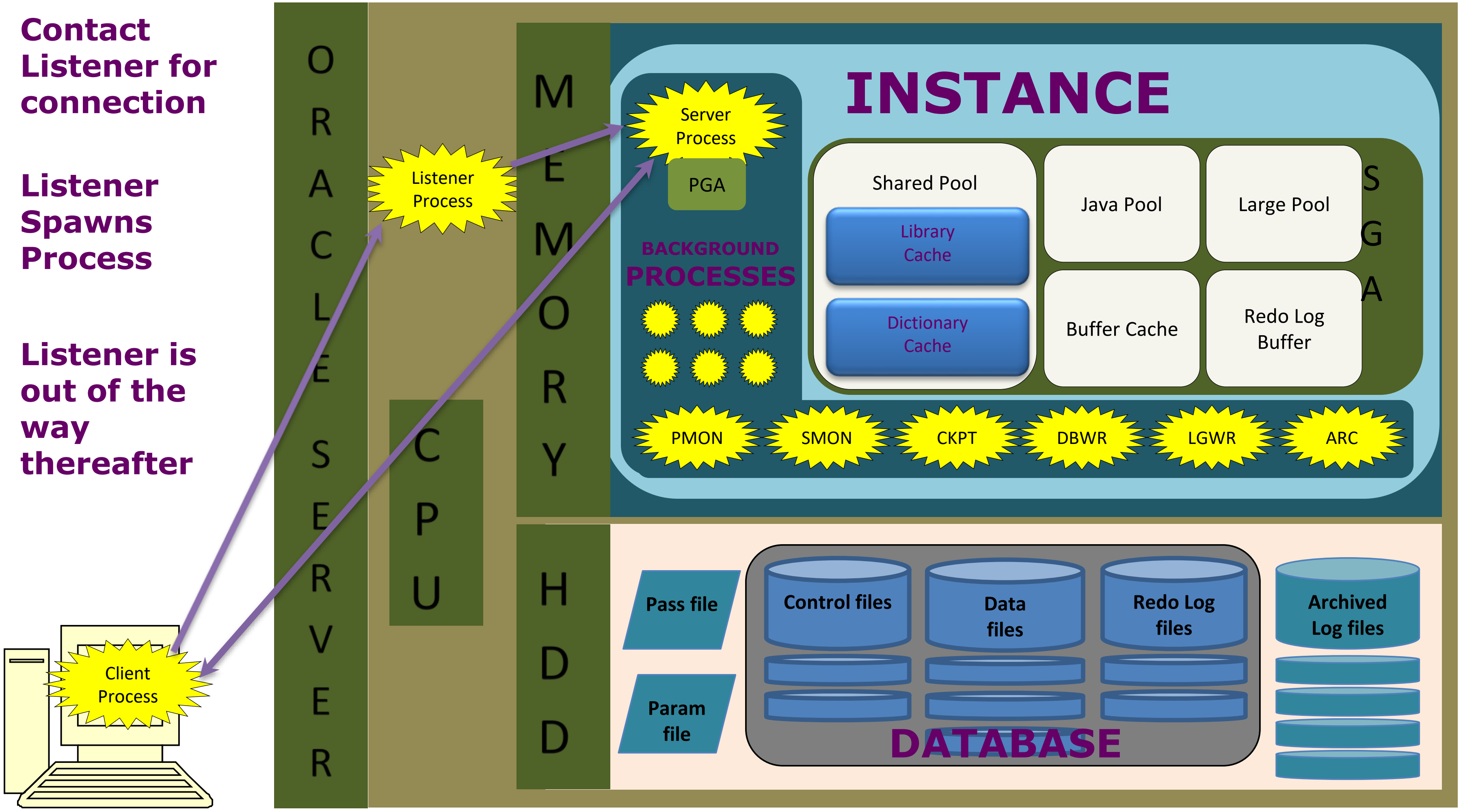


Connection Processing

Contact Listener for connection

Listener Spawns Process

Listener is out of the way thereafter



Connection Processing

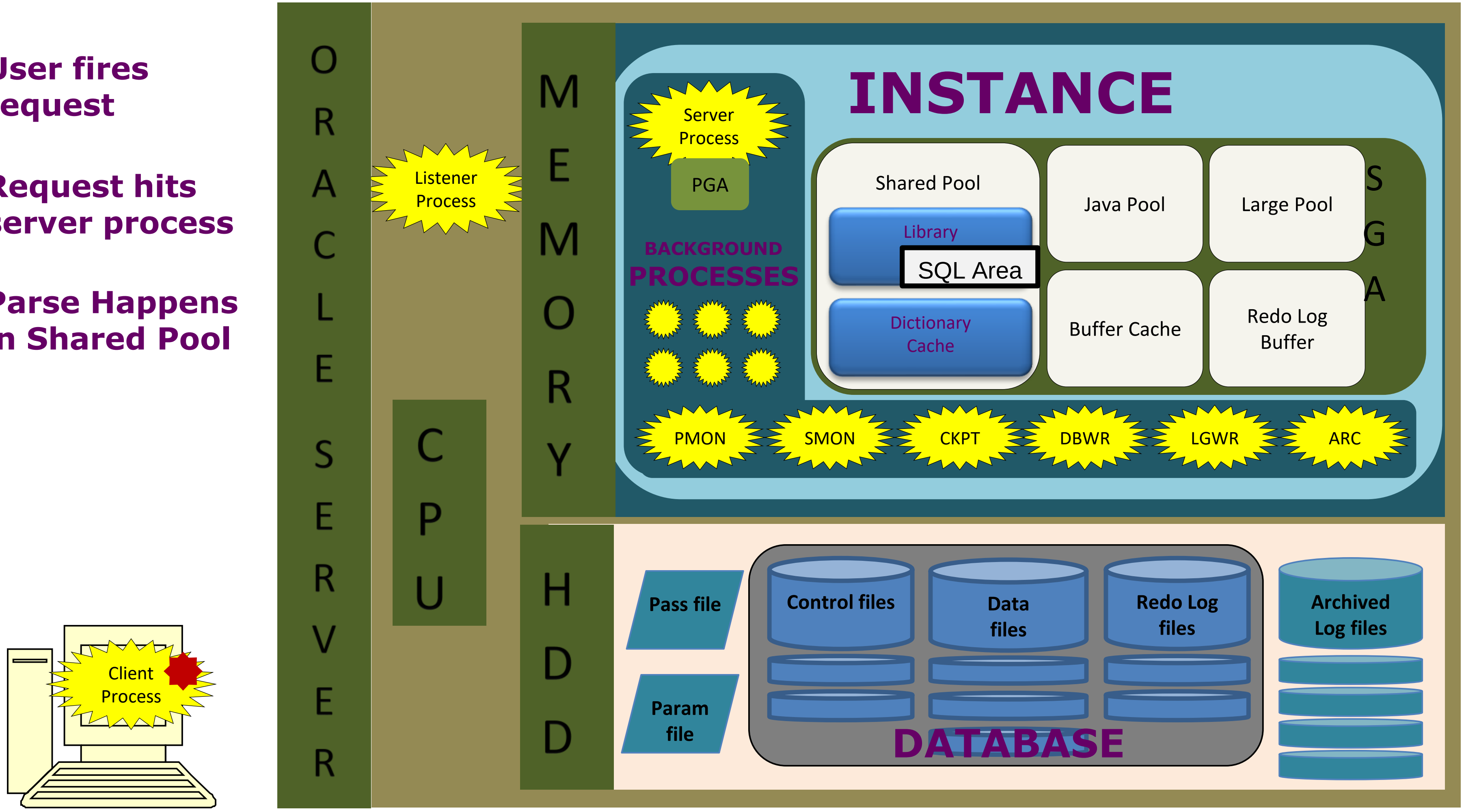
- There is Work Involved in the Database in creating a Connection
- Minimize Connection Requests
- Use Connection Pools

Select Statement Processing

User fires request

Request hits server process

Parse Happens in Shared Pool



Statement Parsing

- Syntax
- Semantics
- Security (privileges)
- Generate Execution Plan

- After Parse

- Bind Variables
- Execution
- Fetch

Parsing

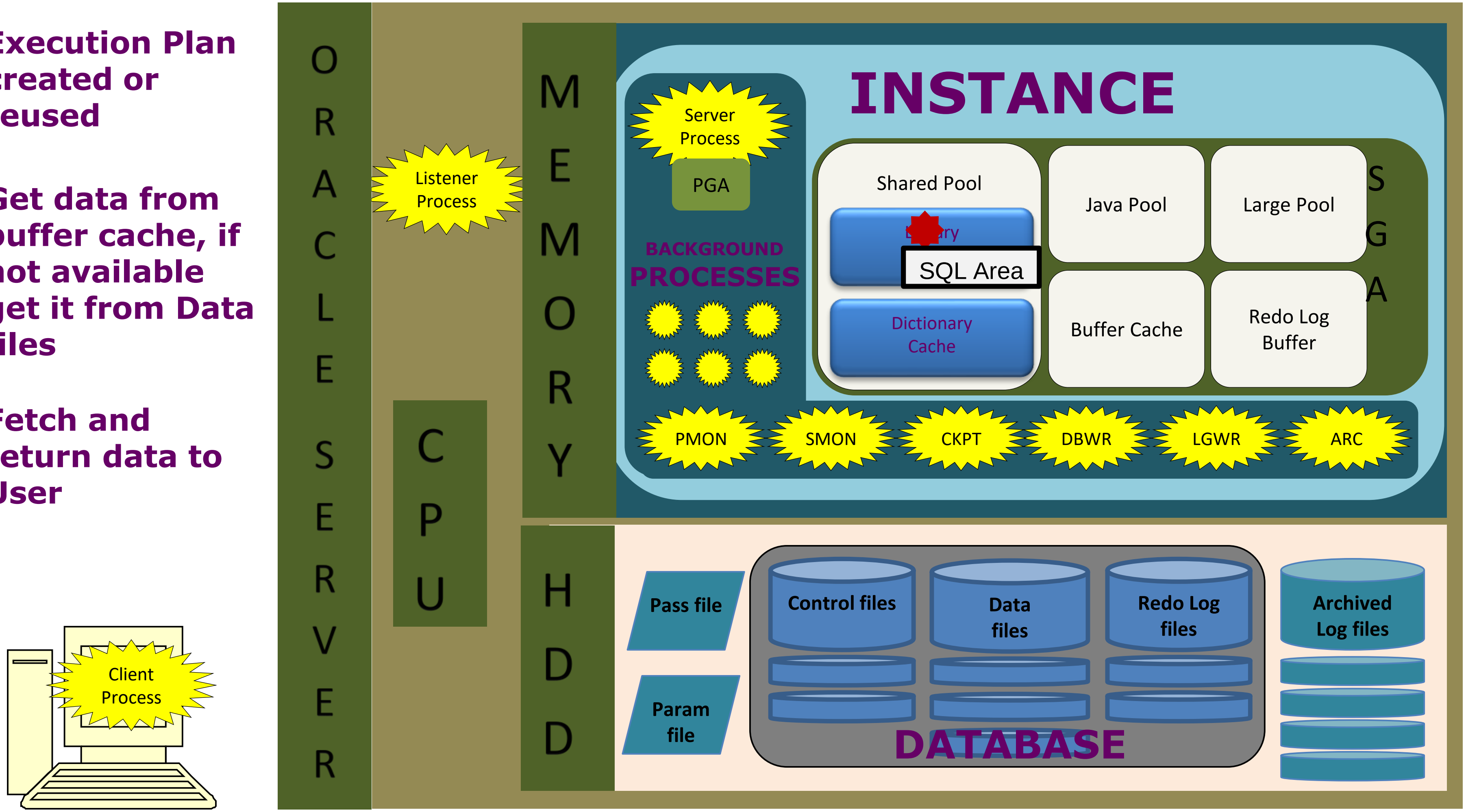
- There is a lot of Work Involved in Hard Parsing
  - Shared Areas in Memory Locked – Latch / Mutex
  - CPU Resources
  - Additional Memory allocation for SQL Area
- There is Work involved in Soft Parsing
  - Search the Library cache for the SQL Area
- Can we do faster than Soft Parse
  - Session_cached_cursors (with Dynamic SQL)
  - Open_cursor
  - These imply more memory for PGA

Select Statement Processing

Execution Plan created or reused

Get data from buffer cache, if not available get it from Data files

Fetch and return data to User



DML Statement Processing

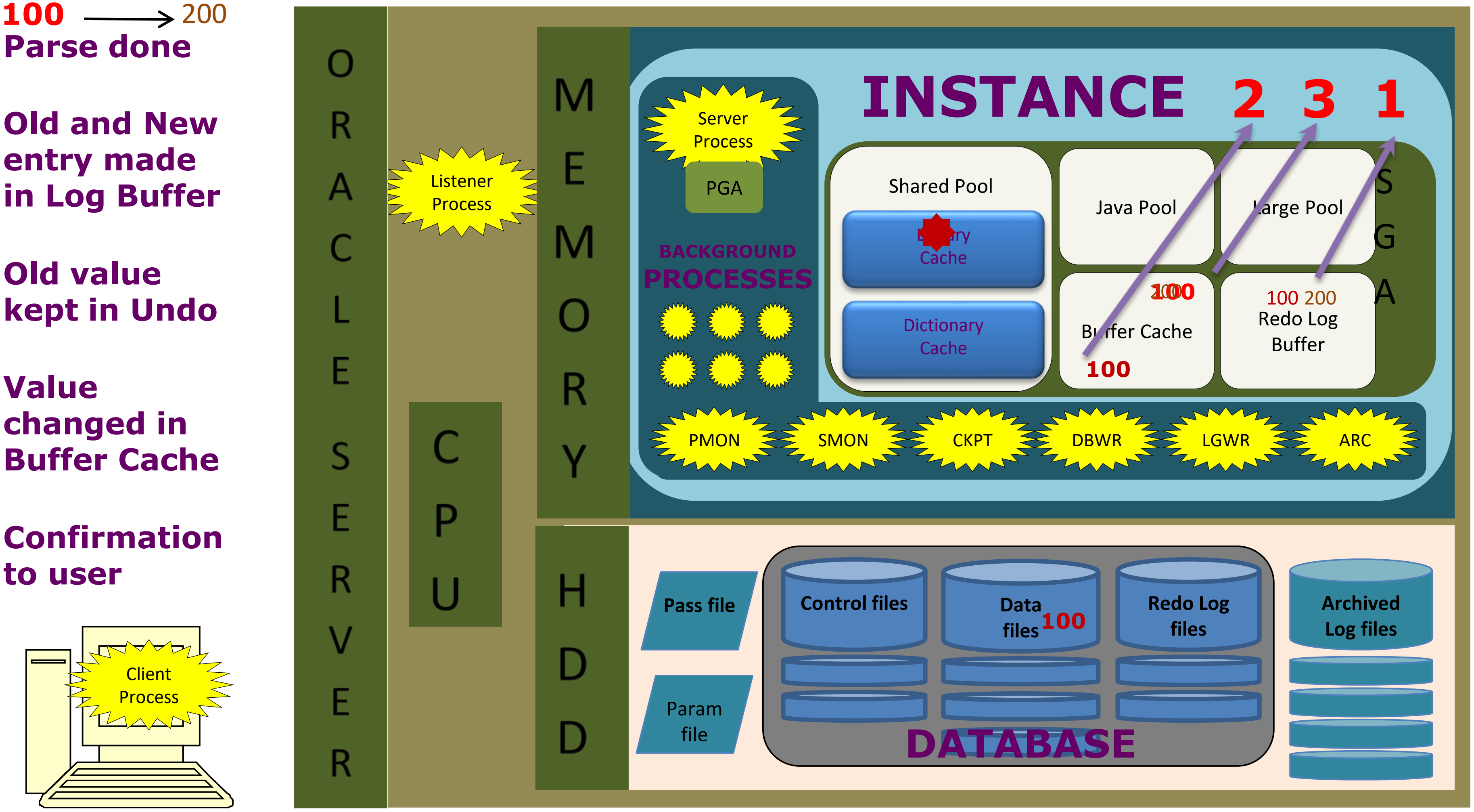
100 → 200
Parse done

Old and New entry made in Log Buffer

Old value kept in Undo

Value changed in Buffer Cache

Confirmation to user



Commit Processing

Generate SCN and insert into Log Buffer

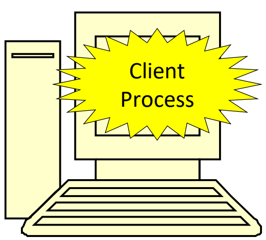
Flush log buffer to redo log files

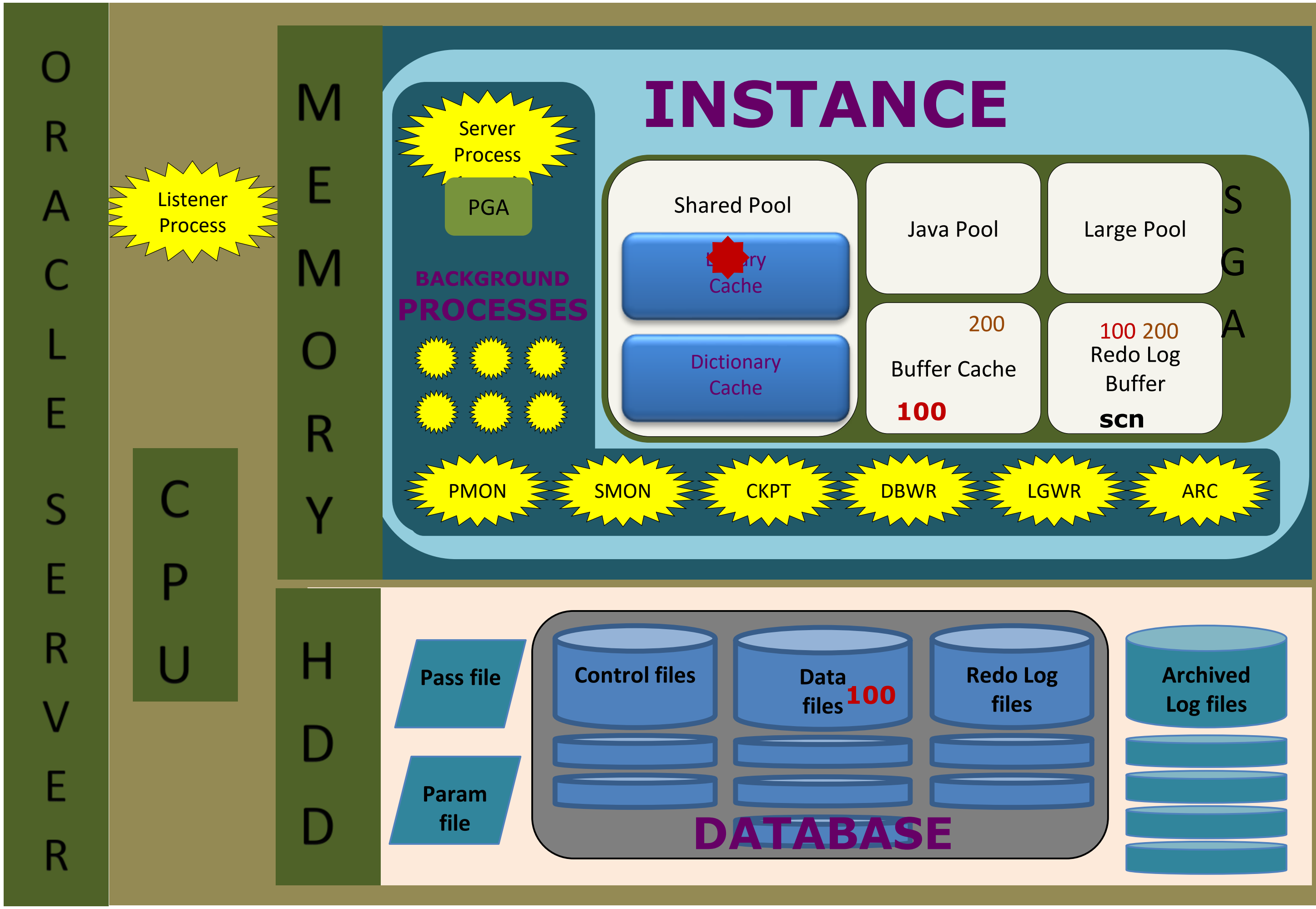
Confirm to user

Unlock old value in Undo

Unlock data

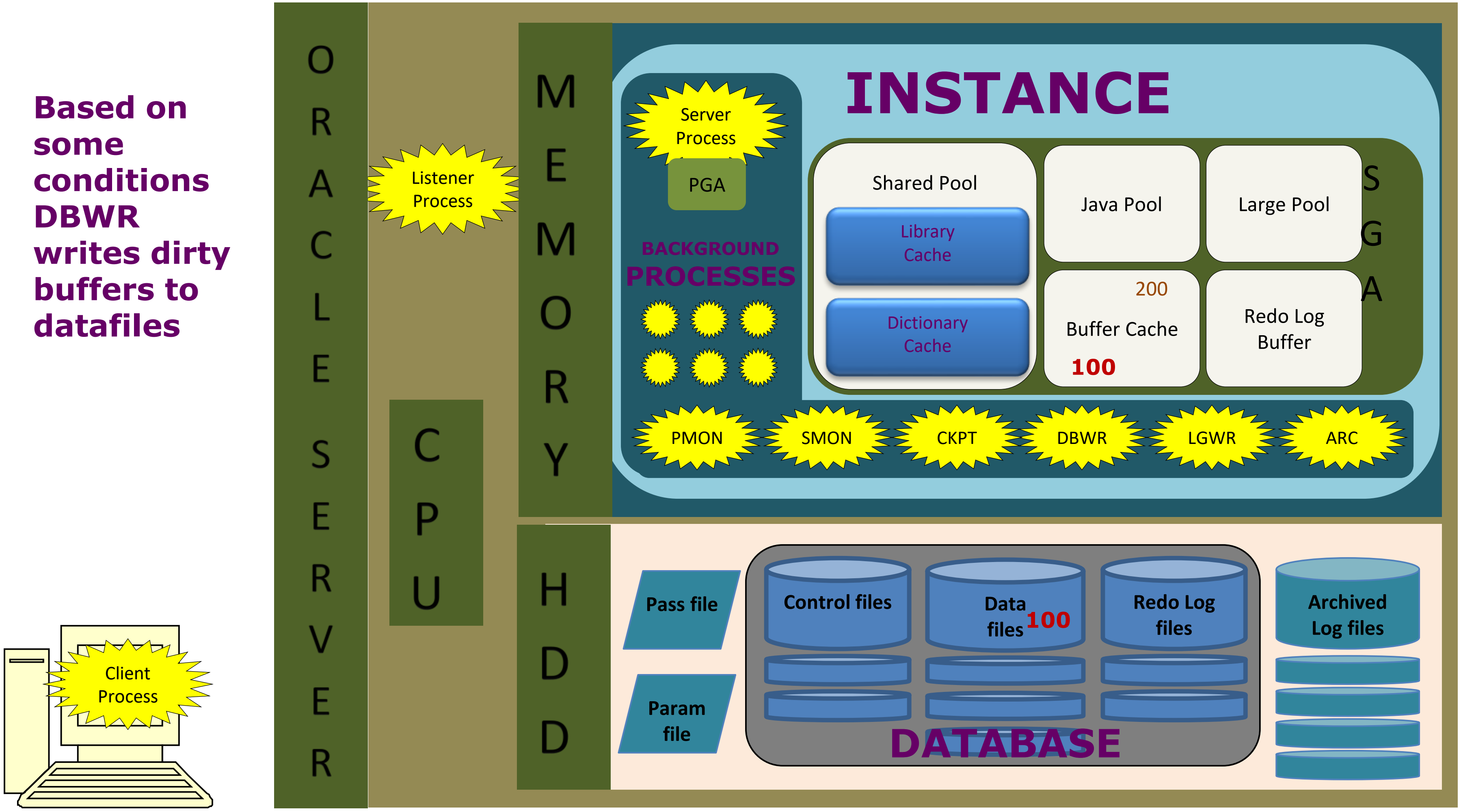
That's it!!!





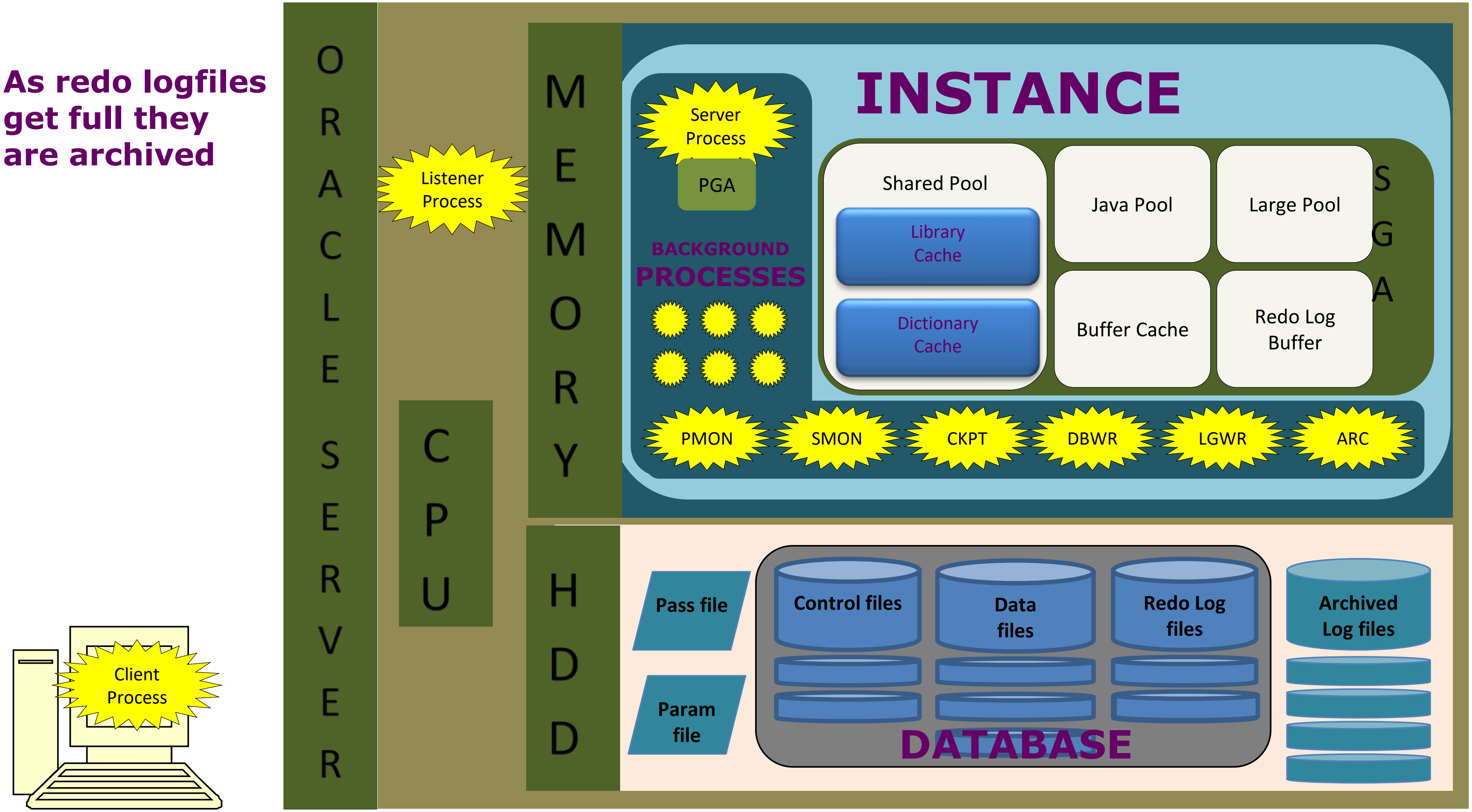
DB Writer

Based on some conditions DBWR writes dirty buffers to datafiles



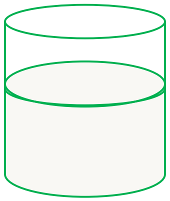
ARChiver

As redo logfiles get full they are archived



How SQL uses the resources

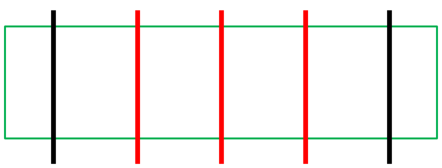
MEMORY



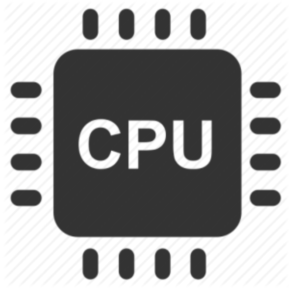
Full / Empty

CPU

Busy



Idle

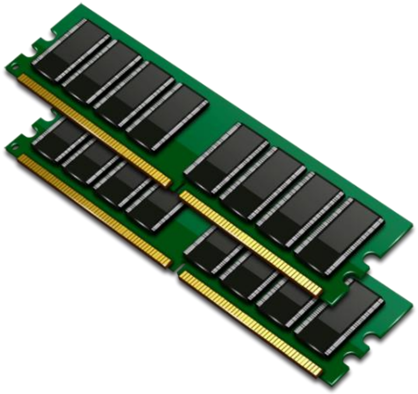


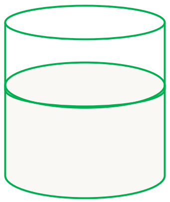
HARD DISK

How SQL uses Memory

MEMORY

- Parsing – Library Cache
- Definitions – Dictionary Cache
- Data fetch – Buffer Cache
- Logging – Redo Log buffer
- Sorting – PGA
- Connections - Processes





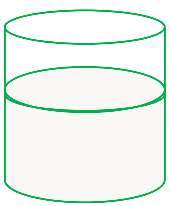
Full / Empty

- Backup – Large Pool
- Java Programs – Java Pool

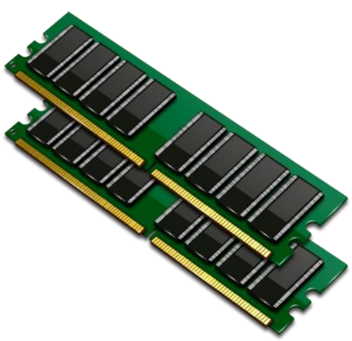
Efficient Memory usage by SQLs

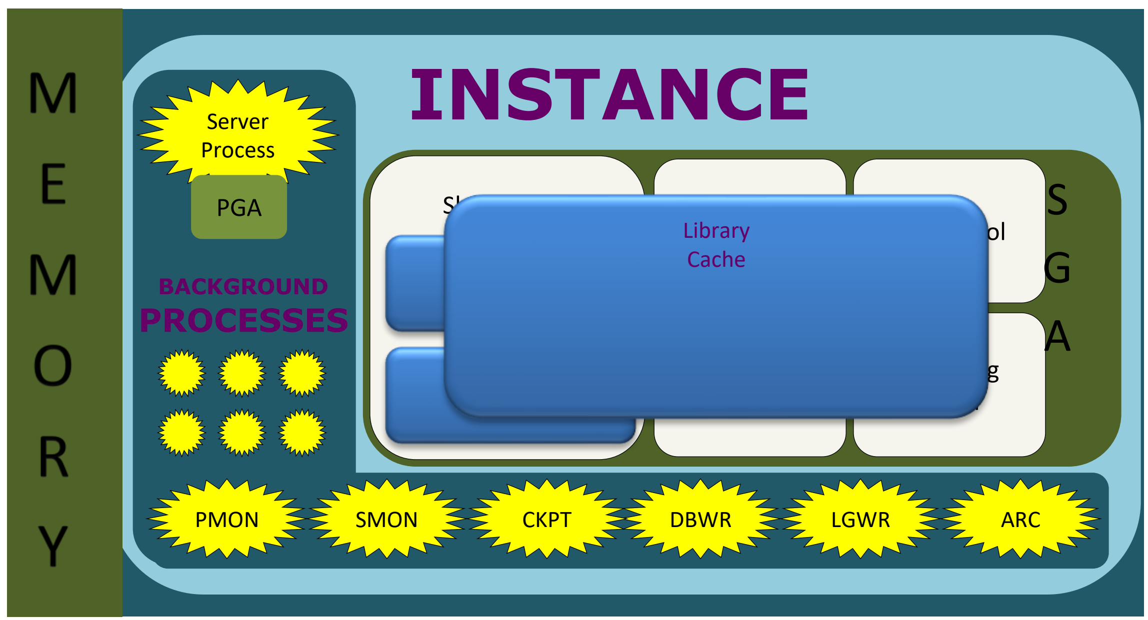
MEMORY

- Library cache – Prepared Statements / Bind variables
- Reduces CPU usage also



Full / **Empty**

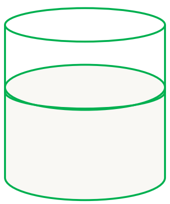




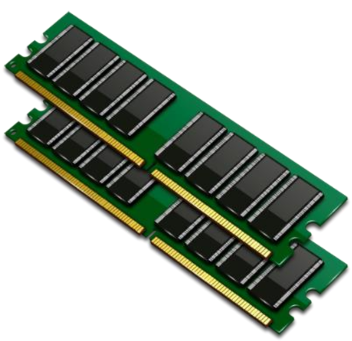
Efficient Memory usage by SQLs

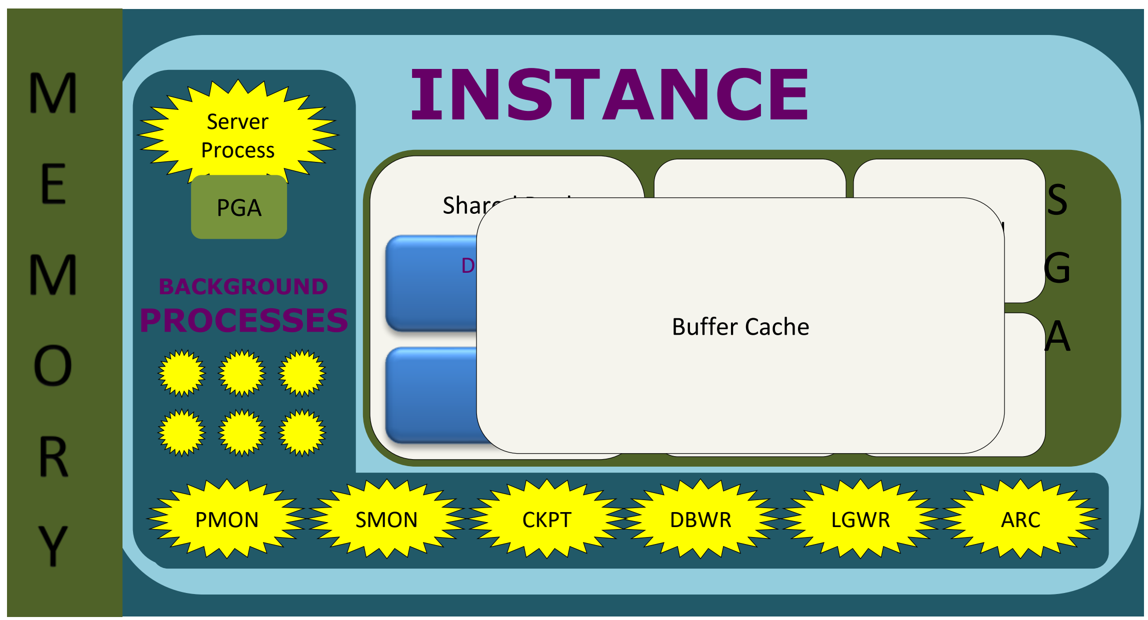
MEMORY

- Indexed where clause- best on Primary Key



Full / **Empty**

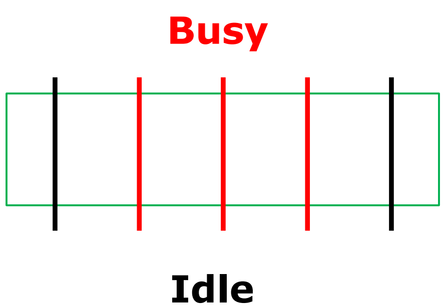


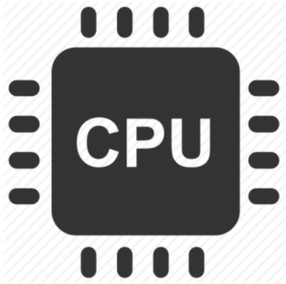


How SQL uses CPU

CPU

- Parsing – Library Cache
- Executing SQL
- Function calls
- Sorting / Hashing

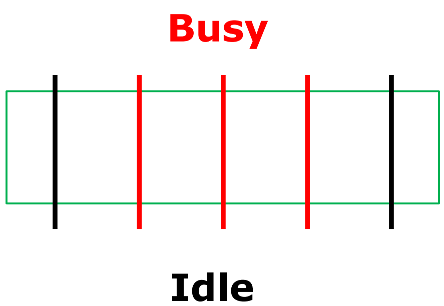


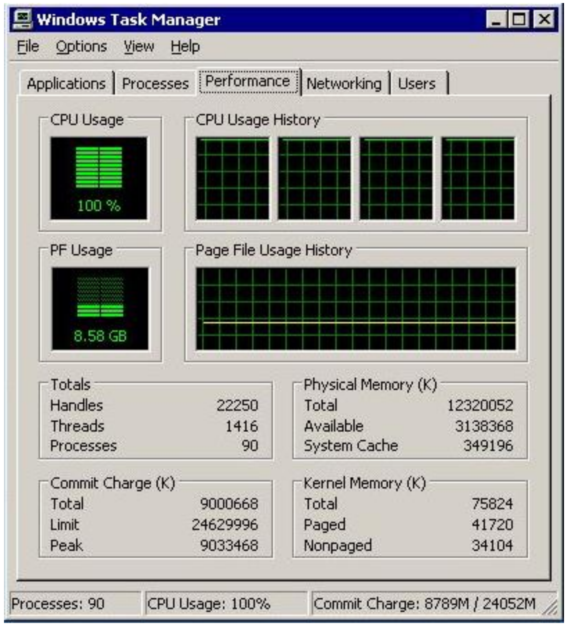


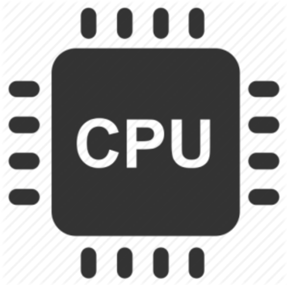
Efficient CPU Usage by SQLs

CPU

- Prepared Statements
- Reduce Function calls
- Avoid Group by / Order By/ Distinct





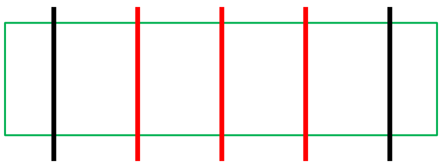


How SQL uses Hard Disk

HARD DISK

- Read from Data files
- Write to Data files
- Write to Redo log files

Busy



Idle

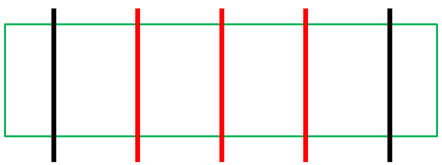
- **Other I/O**
- Write to Control Files
- Write to Archive Logs
- Other Backup related I/O

Efficient Hard Disk usage by SQLs

- Ensure where clause (with Index?)
- Avoid Distinct / Order by / Group by
- Actually Don’t visit the Disk At All…. ???

HARD DISK

Busy



Idle

Labs

- Cursor Caching in sessions
- Growth of Cursors in Library Cache

Summary

- How does Oracle Database use its Resources
- Memory , CPU and I/O are used and can be optimized

Understanding the Importance of Design

Chapter 2

Design Fundamentals Specifically for Performance

Which is the Best Design?

Or is it This

Is there a right Answer???

Well it Depends!!!

Agenda

- First of all Data Model
  - Entity Relationship
  - Dimension Model
- Data Types of Attributes
- Column Order
- Using Constraints
- Connection Management
- Alternate Storage Techniques
  - Discussed in Physical Design
- Where to do the Work (Application Design)

Data Model

- Best is to Model the Entire Application
- Have it documented… Do you have?
- ER Modelling – Specifically for OLTP
  - Integrity and Consistency
  - Avoid Redundancy
- Dimension Modelling – Specifically for DWH
  - Sources take care of Integrity
  - Keep Redundancy to increase Performance
- Use a Tool for Database Design

Data Types of Attributes

- Use the Right Data type for Attributes / Columns
- Why not store Numbers in Varchar2
- Why not store Date in Varchar
- Anyway I can have a Function to do Conversion if required
  - Do work to get that… which could have been avoided

Problem with Wrong Data Types

- Can we Validate?
- Loss of Information (Eg. Time Zone into Date)
- Loss in Functionality (Dates not ordered properly)
- Optimizer is wrongly Informed / Decides Wrongly

Column Order

- Does it Really Matter
- Anyway the Blocks are read fully … there is no way to read a column…(in Buffer Cache.. Not INMemory)
- Even After reading a Block… Picking the Row… There is work involved in getting a Column
- Also, let applications not Select * or Columns now wanted in the Application

Using Constraints

- Is it necessary to Implement Constraints in the Database
  - Primary Key, Foreign Key, Unique, Check, Not Null
- Its very much possible to Implement them in Application
- Will there be a need for a New Application?
- Will the data be modified directly in the Database
- Is the optimizer knowing about the Data, to identify the best Execution Plans?

Using Constraints

- Primary Key and Unique Constraints provide Index Lookup preference when those Columns appear in Predicate
- Not Null can perform Fast Full Scan on Indexes
- Foreign Key can help in Joins
- Optimizer gets information from Check Constraints

Where to do the Work (Application Design)

- Database will be used to Store Data… That’s it???
- Do everything else from the Application
  - Did you think about Network Overhead
  - Data Processing is best in the Database
  - Indexes, Alternate Storage Techniques Optimization
    - Materialized Views
    - Partitioning
- Workshop on Where to do what

Lab

- Using Right Data Type
- Column Order
- Using Constraints

Summary

- Spend time in Logical Design - Data Model
- Identify Correct Data Types of Attributes
- Know Application Access Patterns - Column Order
- Implement Constraints
- Connection Management
- Identify the Best place to do a piece of Work

Will Adding Hardware Solve Performance Issues?

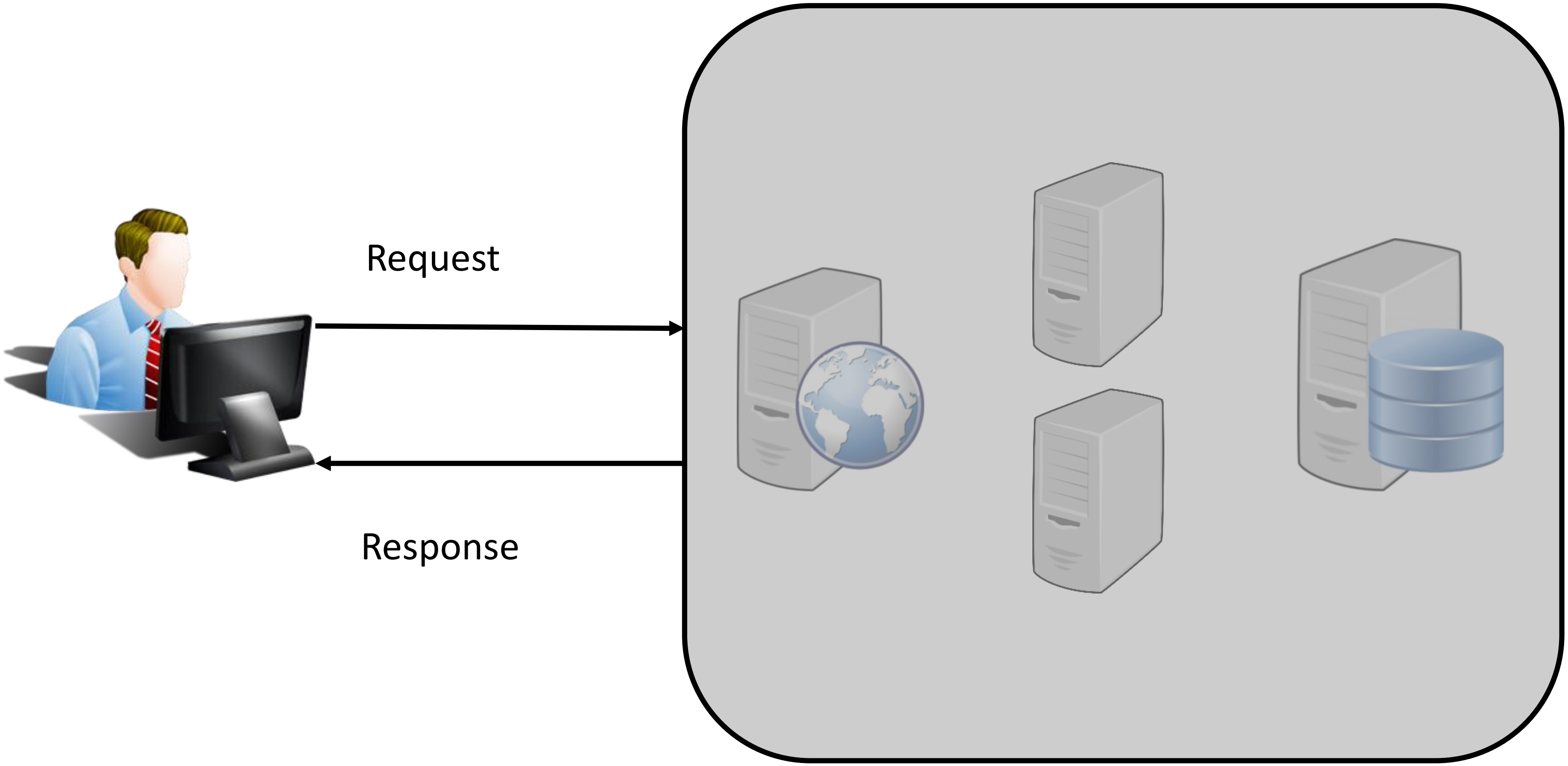
Chapter 3

Agenda

- Understand
  - User Response Time
  - Application Response Time
  - Database Response Time
- What is DB Response Time Made up of

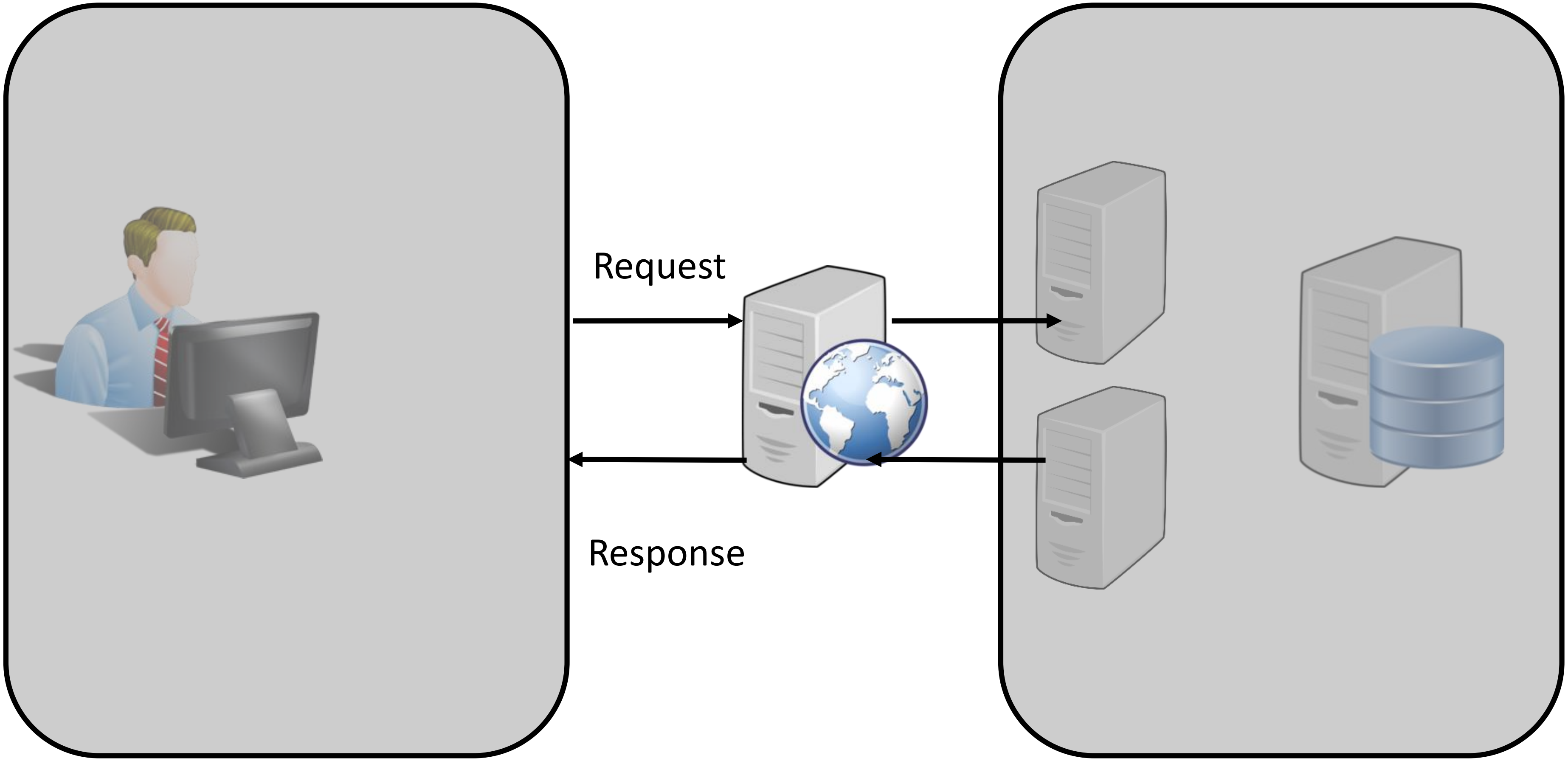
What is User Response Time?

User Response Time



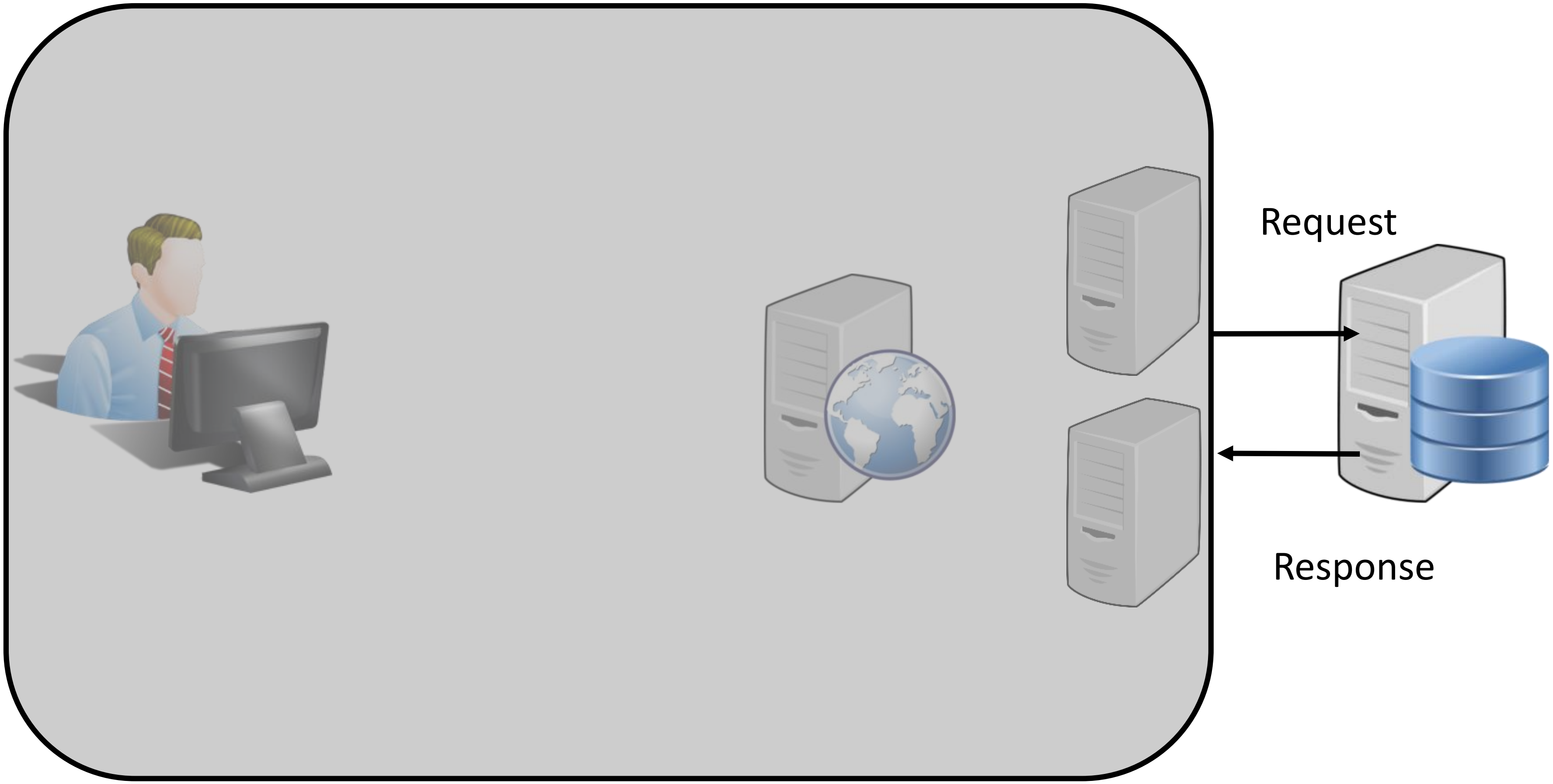
Total Time taken to get back the response after a click by the User

Application Response Time



Time taken by Application to send back the Response after receiving Request

Database Response Time



Time taken by Database to send back the Response after receiving Request

Do you Measure?

- End User Response Time
- Application Response Time
- Database Response Time

- Do you know how to?

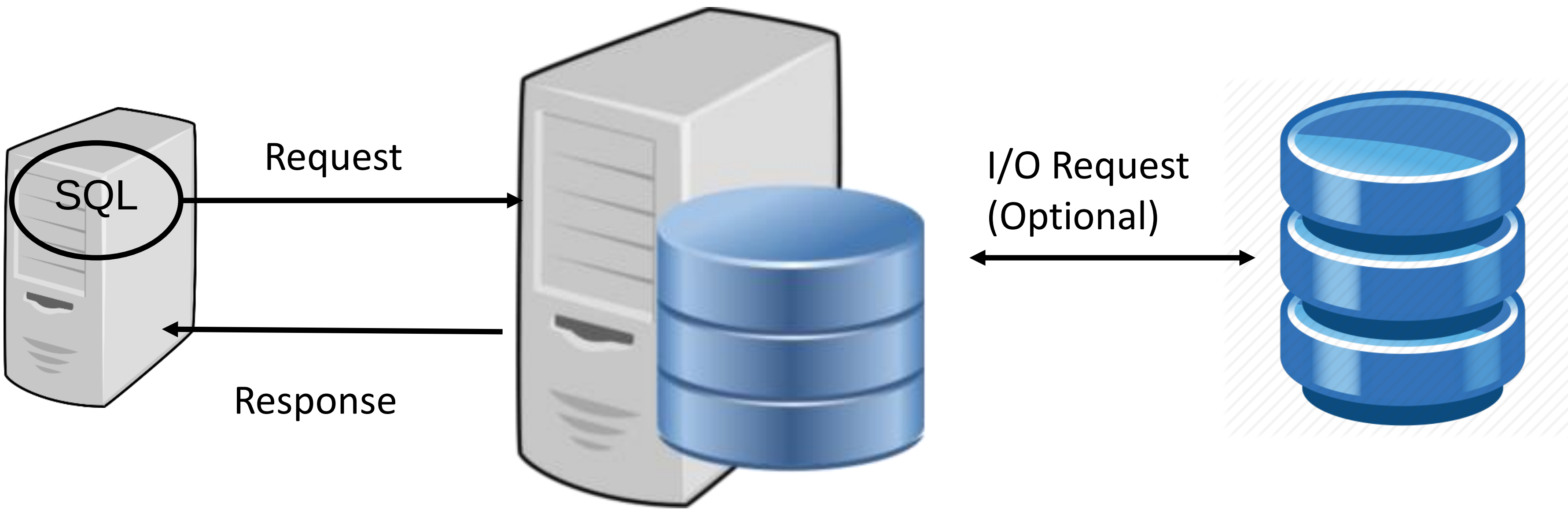
If you are Measuring, is it Good?

- What is Good?
- Do you have **Non-Functional Requirements** in place
  - Performance SLAs
  - Data Volume in tables
  - Workload pattern for Actions/ Transactions
  - Security
- If you have SLAs only then you know if something is Good or Bad
- Test and Benchmark your software for the above

What happens in Database

PARSE
EXECUTE
PROCESS
SORT etc.

Perform
I/O
From
Storage



What is Database Response Time

- SQL Response Time / SQL Elapsed Time
  - Helpful for SQL Tuning

- SQL Execution = CPU time + WAIT Time
- Why Wait
  - Lack of Resources
  - Slow Resources
  - Application Locking
  - Latching
  - I/O Overhead…..

Overall Database Activity – DB Time

- Sum of All Executions in Database
- Sum of All CPU used + Sum of All Waits

- Helpful for Database/Instance Tuning (Overall)

Can Hardware Improve the Performance

- What is Hardware
  - CPU
  - Memory
  - Network
  - Storage
- Is the Hardware a Problem?
  - Is your Database Slow because of Lack of it?
  - What is your Database Waiting for?
  - What is the resource crunch the database is having?

Summary

- Understood the concepts of
  - User Response Time
  - Application Response Time
  - Database Response Time
- What is DB Response Time Made up of

Accurately Identifying Performance Metrics

Chapter 4

Agenda

- Database Performance Data
- Wait Events
- Ratio Analysis

Database Performance Data

- Database Statistics
  - Wait events
  - Time Model
  - System and Session
- Metrics
- ASH (Sampled)
- Ratio Analysis
- AWR

Database Statistics

- Captured in V$ Views
- Captured Automatically
- Different Views have Different intervals of Refresh
- Many depend on TIMED_STATISTICS and STATISTICS_LEVEL parameter
  - TIMED_STATISTICS = TRUE/FALSE
  - STATISTICS_LEVEL = BASIC / TYPICAL / ALL

- OS Statistics
  - v$osstat
  - Better use an O/S level Capture

Wait Events

- Process Prevented from doing an activity, it Waits
- What is the reason for a Process to wait?
  - Application Locking
  - I/O
  - Latch Waits – Shared Pool
  - Buffer Chain Latch – Buffer Cache
  - Many More

Wait Metrics

- V$EVENT_NAME – List of all Wait Events in Database

- V$SESSION_WAIT – Session just waited or waiting
- V$SYSTEM_EVENT – Total for all Session for all Waits
- V$SESSION_EVENT – All Waits for all Sessions

Wait Classes

- Waits grouped into Smaller list of Classes
- Easy to diagnose what is the Major Reason for Waits
  - Administrative – DBA Tasks
  - Application – User Application Code
  - Cluster – RAC specific
  - Commit – Log File Sync Wait
  - Concurrency – Database Resource Contention eg. Latches
  - Configuration – Inadequate Sizing – Buffer Cache, Log Size
  - Idle - Sessions not doing anything – Inactive
  - Few more are there

Generally Encountered Waits

- Buffer Busy Waits
- Cursor : Mutex S – Shared / X – Exclusive
- Db file * Read / Write
- Direct Path Read / Write
- Free Buffer Waits
- Log Buffer Space
- Log File Sync

Read Waits

- On a Well Tuned System… You will See This
- Database is doing a lot of Reads
  - Check TOP SQL Section
- Reads are slow from the I/O Substystem
  - < = 20ms waits is acceptable I/O performance
  - Check Tablespace I/O Stats section

Should you worry if you See Waits

- What % of DB Time is the Wait Contributing ?
- Is the SLA being Met
  - Yes – Then you don't need to worry…
  - No – Dig Deeper for the reason for the Wait
- If Yes, above, may be it's a checklist entry for Proactive Tuning

SYSTEM and SESSION level

- V$ Views available for both
- AWR report provides SYSTEM level between 2 snaps
- ASH report drills into details of Waits for the period
  - Session Wise
- You also have Service Level Waits – V$SERVICE_EVENT

Metrics

- Rate of Change in Cumulative Statistics
- V$METRICNAME – Lists the Various Metrics Captured
- V$METRIC – Lists the recent metrics captured
- V$METRIC_GROUP – Interval for Computation and History retention

Ratio Analysis

Memory Ratio Analysis

- V$ Advisories
  - V$DB_CACHE_ADVICE
  - V$SHARED_POOL_ADVICE
  - V$JAVA_POOL_ADVICE
  - V$PGA_TARGET_ADVICE
  - V$SGA_TARGET_ADVICE
  - V$MEMORY_TARGET_ADVICE

- All of these are based on Ratio Analysis

Buffer Cache Ratio Analysis

- Three Statistics from V$SYSSTAT
  - consistent gets from cache
  - db block gets from cache
  - physical reads cache
- Buffer Hit Ratio

1- (Physical Reads / (consistent gets + db block gets))

Shared Pool Related Statistics

- Library Cache – V$LIBRARYCACHE
  - Reload – amount of Re-parsing an SQL that aged out
  - Invalidation – Typically DDL or Statistics Gathering
  - Library Cache Hit – Objects found in Memory%
    - SUM(PINHITS)/SUM(PINS)

Shared Pool Ratio Analysis

- Increasing Shared Pool increases
  - Library Cache
  - Dictionary Cache
  - Result Cache

I/O Ratio Analysis - Reads

- Statistics from V$SYSSTAT
  1. Physical Read Total I/O Requests
     - Includes all Activities
  2. Physical Read total Multi Block Requests
     - Subtract this from first to see single block requests total
  3. Physical read I/O Requests
     - Read into Buffer Cache and Direct Reads
     - Subset of 1
  4. Physical Read Bytes
     - IO in Bytes for Application Activity
  5. Physical Read Total Bytes
     - Total IO for all Activities in Instance
     - Difference from 4, is non application activity

I/O Ratio Analysis - Reads

- Statistics from V$SYSSTAT

  6. Physical Reads – Made up of

     - Physical Reads Cache – From disk to Buffer Cache
     - Physical Reads Direct -
     - Includes all Activities

I/O Ratio Analysis - Writes

- Statistics from V$SYSSTAT
  1. Physical writes and Physical Write Bytes
     - Data Blocks written to Disk due to Application Activity
     - Includes – Physical Writes Direct and from Cache
  2. Physical write IO requests
     - Buffer Cache and Direct Load writes
  3. Physical write Total IO requests
     - All activities including Application
  4. Physical Writes Non Checkpoint
     - Overhead due to FAST_START_IO_TARGET
     - Subtract from Physical Writes to know the Overhead

Ratio Analysis - IO

- Using the Statistics Given in the last 2 slides
- Figure out How much Direct IO happening
- How much IOPS and MBPS is happing in your database
- Same statistics are available under Instance Activity Statistics in AWR (11g)
- In 12c you have a key Instance Activity section
- Figure out is your database I/O bound…

Key Instance Activity Stats

- Ordered by statistic name

Statistic	Total	per Second	per Trans
db block changes	1,359,547	75.47	9.33
execute count	792,800	44.01	5.44
logons cumulative	2,748	0.15	0.02
opened cursors cumulative	193,953	10.77	1.33
parse count (total)	360,739	20.03	2.48
parse time elapsed	891	0.05	0.01
physical reads	66,133,121	3,671.34	454.06
physical writes	7,228,468	401.28	49.63
redo size	300,502,480	16,682.22	2,063.22
session cursor cache hits	408,220	22.66	2.80
session logical reads	91,960,618	5,105.14	631.39
user calls	1,097,571	60.93	7.54
user commits	28,858	1.60	0.20
user rollbacks	116,789	6.48	0.80
workarea executions - onepass	48	0.00	0.00
workarea executions - optimal	29,023	1.61	0.20

Summary

- Database Performance Data
- Wait Events
- Ratio Analysis

Understanding Physical Design

Chapter 5

Agenda

- Result of Data Modeling
- Optimizing SQL Execution
- Types of Indexes
- Alternate Storage techniques – Physical Design

Result of Data Modeling

- List of Tables
- Attributes and Keys Identified
- Right Data Types for Attributes
- Relationships between Entities

- Next Step is to create
  - Tables
  - Constraints

Optimizing SQL Execution

- At a bare Minimum SQL access Data
- Data is in Tables.. In Data blocks on Data Files
- Optimizer Does Many things for
  - Order to Access the Tables (Data)
  - Identify How to Access these tables (Access Paths)
  - How to join the Row Sources
- Finds the most efficient way to do all that

- Your SQL defines
  - What tables to Access and Join
  - What to do with the Columns – Function Calls
  - Sort / Distinct required… Use PGA
  - Aggregation

Optimizing Data Access

- SQL Accesses Data - Thus you need to
  - Optimize Data Access to Optimize Execution
  - Only then Optimizer can find best paths

Example

- How many blocks required to Access 100 Rows?
- How many rows in a Block?

- Objective – **Minimize Block Visits**

Pack Rows Tightly in Blocks

- Block Size
- PCTFREE and PCTUSED (MSSM)
- INITTRANS (for Concurrency)

- When should we not pack rows Tightly?

Block Size – Case Study

- Table has – 1,000,000 Rows
- Each Row need 1kb space
- How to Store it ?

- No of Blocks Required (with blocks tightly packed)
  – 4k BlockSize - 250,000 Blocks
  – 16k BlockSize – 62,500 Blocks
- Guaranteed Default Concurrency – INITTRANS 2
  – 4k Blocks – 500,000 rows
  – 16k Blocks – 125,000 Rows

Alternate Storage Techniques

- Heap Tables… Normal Tables
- Remember … the Different Vehicles
  - Which is Best for performance?
- Consider
  - Indexes (Again different Types available)
  - Partitioning
  - Materialized Views
  - Index Organized Tables

Types of Indexes

- BTree and Bitmap Indexes
- Normal and Reverse Key Indexes
- Index Partitioning
- Compressed Indexes
- Unique and Non-Unique Indexes
- Indexes on Not Null Columns
- Covering Indexes
- Index Partitioning

Partitioning

- VLDB Scenarios
- Sometimes can be useful on Smaller Environments
- Range or Hash or List or Composite
- What indexes to Create

Index Organized Tables

- When a table is always accessed with Predicate on PK
- Number of Columns in IOT and Overflow Segments

Materialized Views

- Run Time Computation Vs Pre-Compute Data
- Query Re-Write
- Stale Tolerated

Summary

- Result of Data Modeling
- Optimizing SQL Execution
- Types of Indexes
- Alternate Storage techniques – Physical Design

Lab

- Tightly Packed Rows
- Using IOT
- Using Reverse Key Indexes
  - Real World Performance
  - https://apexapps.oracle.com/pls/apex/f?p=44785:24:116079504583996:PRODUCT:::P24_CONTENT_ID,P24_PREV_PAGE,P24_PROD_SECTION_GRP_ID:9567,141,1745
  - https://youtu.be/wQNPQGUwjbs

Optimizing Storage Configuration

Chapter 6

Agenda

- Understanding Hardware Evolution
- Understanding CPU/ Storage Evolution
- What do you ask when you want more Storage
- Using Compression
- Tiered Storage

Understanding Hardware

- All data needs to be READ into Memory to Process
- The speed of reading depends on the Speed of the Disk Subsystem
- Understand your Storage Specs to know what it is capable of

Evolution of CPU / Processor

- 1970 – 375 kHz
- 1975 – 1 MHz
- 1989 – 25 MHz
- 1997 – 300 MHz
- 2006 – 2.6 GHz (2 Cores)
- 2016 – 3.6 GHz (16 Cores)

- It's a mixed evolution… Not a single Vendor

Evolution of Storage

- 1956 – IBM RAMAC 305 – 5MB – Size of a Refrigerator
- 1992 – Seagate – 7200 RPM – 2.1 GB
- 1996 – Seagate – 10000 RPM
- 2000 – Seagate – 15000 RPM
- Today the Talk is about SSDs / Flash

Evolution of Exadata As an Example

Per Server	DB server CPU	DB Server Memory	Storage HDD	Storage Flash Cache
X2	2 * 6 Cores	144 GB	12 * 600 GB HP (15000 RPM)	396 GB
X3	2 * 8 Cores	256 GB	12 * 600 GB HP (15000 RPM)	1.6 TB
X4	2 * 12 Cores	512 GB	12 * 1.2 TB HP (10000 RPM)	3.2 TB
X5#	2 * 18 Cores	768 GB	8 * 1.6 TB Flash	6.4 TB
X6#	2 * 22 Cores	1.5 TB	8* 3.2 TB Flash	12.6 TB

X2- 2012 , X3-2013 , X4- 2014 , X5- 2015 , X6 - 2016

-Disks are considered as High Capacity as High Performance stopped and Extreme Flash Offered

Evolution of Exadata As an Example

Full RACK	Disk IOPS	Flash IOPS	DISK MBPS	FLASH MBPS
X2	28000	1,500,000	18 GB	68 GB
X3	50000	1,500,000	25 GB	100 GB
X4	50000	2,660,000	24 GB	100 GB
X5#	36,000	4,144,000	25 GB	263 GB
X6#	36,000	4,950,000	25 GB	350 GB

X2- 2012 , X3-2013 , X4- 2014 , X5- 2015 , X6 - 2016

* -Disks are considered as High Capacity as High Performance stopped and Extreme Flash Offered

What do you ask when you want Storage

- Space in GB /TB
- How soon you want

- Do you know the performance Metrics
  - IOPS
  - MBPS

Why do we need Different Tablespaces?

- Avoid Contention?
- Better Manageability?
- Backup Optimization?
- Easier Maintenance?

SPACE vs IOPS vs MBPS

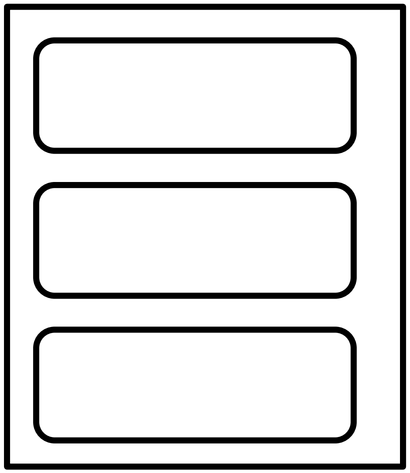
- HDD ~ 250 IOPS
- SSD start at 8600 IOPS and go upto Million IOPS

- HDD MBPS – 260 mbps (Maximum)
- SSD – 1700MBPS PCIE even more

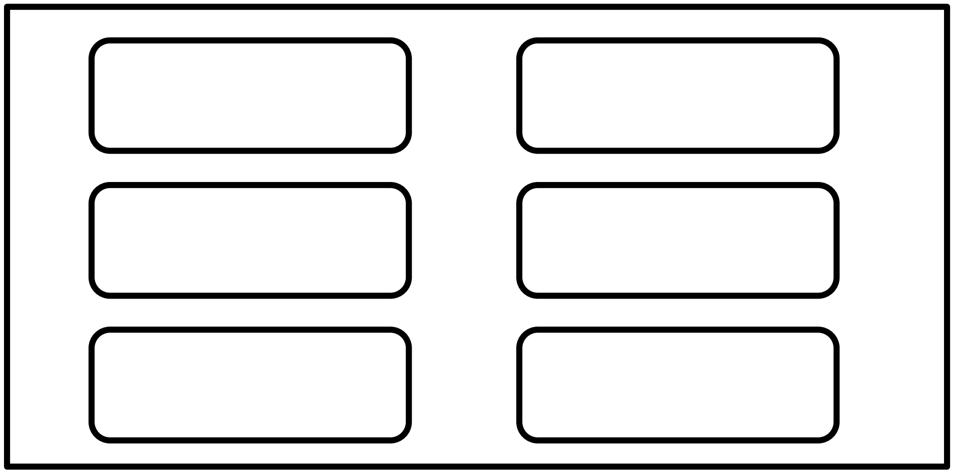
- Larger the DISK … you need to read all the data with the same speed
- Do you know what your LUN is made up of?

What is your LUN Made up of

- How Many disks / Spindles behind it
- Is it from HOT or COLD areas (HDD)
- That will define its performance



LUN from 3 Disks



LUN from 6 Disks

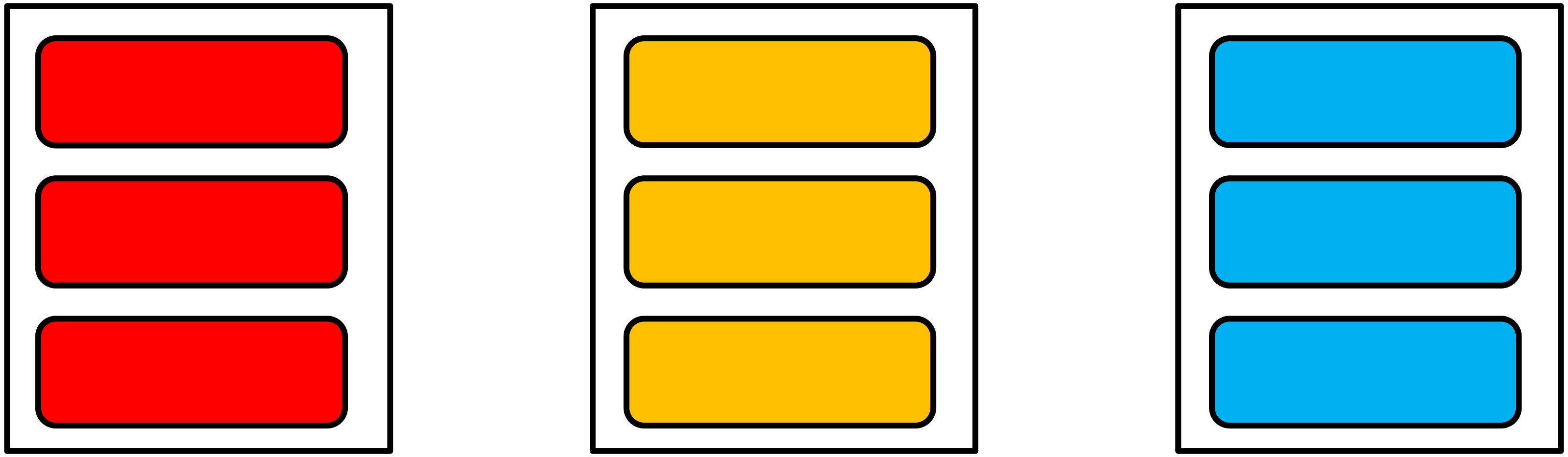
Using Compression

- Reduce Storage and Reduce I/O
- Need to have Spare CPU to do compress / decompress
- Nature of Data and Compression Algorithm determines Compression Ratio
- Choose Wisely
  - Basic
  - Advanced (OLTP)
  - HCC – Query and Archive (Low and High)

TIERED Storage

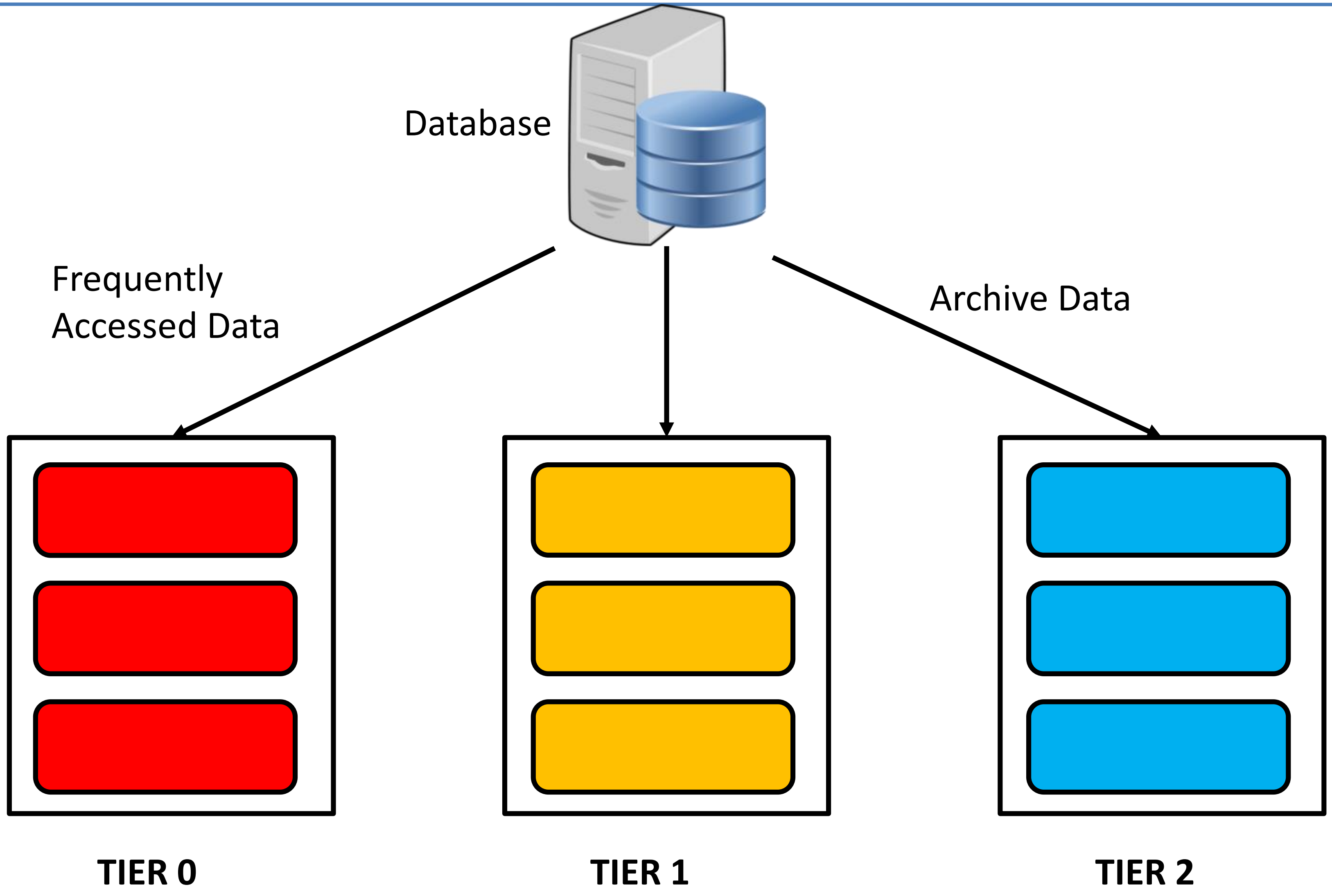
- Today its common to have Different Types of Storage

Fastest Media Costly Restricted	Average: Sped, Cost and Availability	Slowest Media Cheapest Abundant



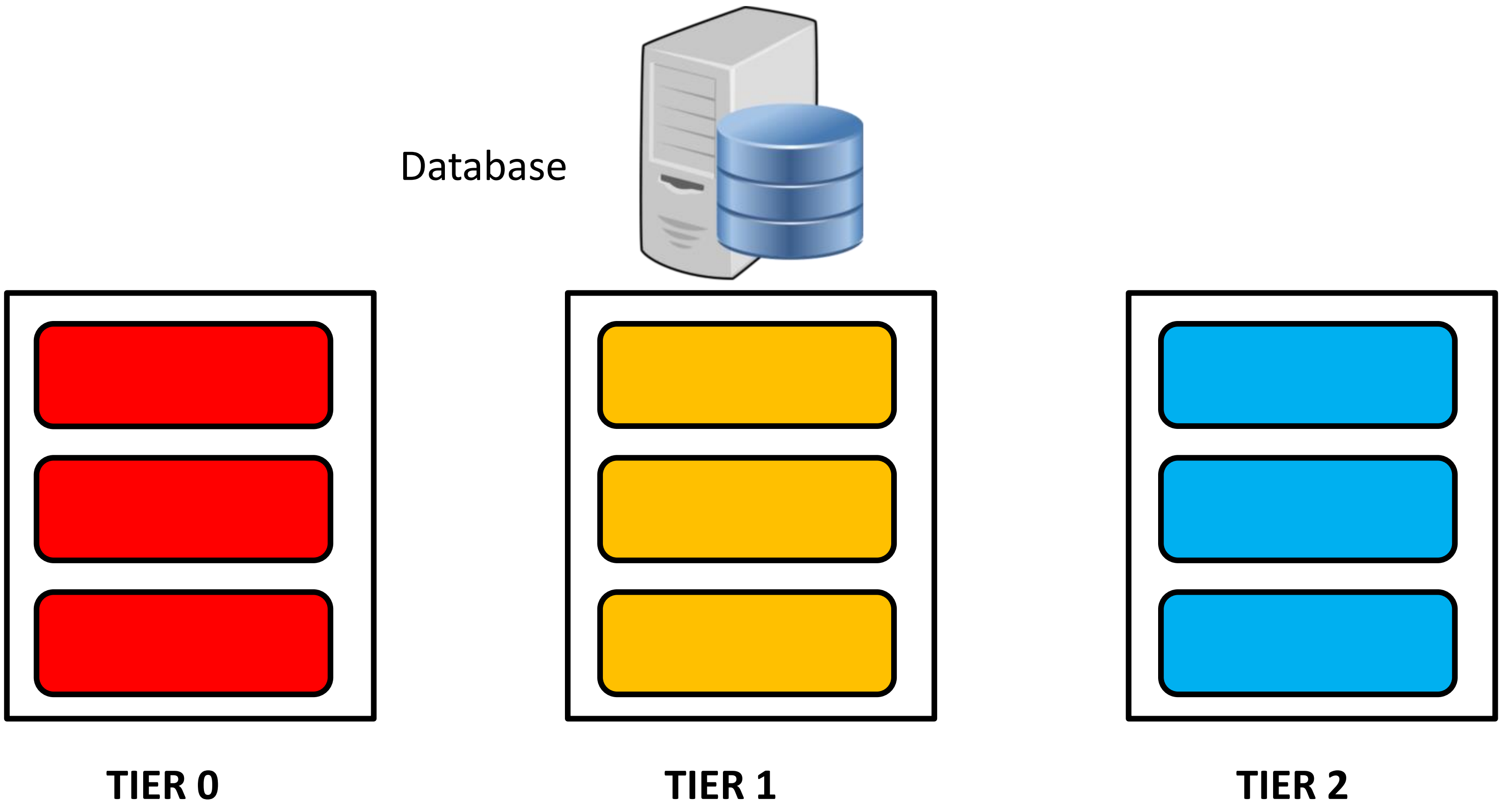
TIER 0	**TIER 1**	**TIER 2**

Using TIERED Storage- DBA Manages

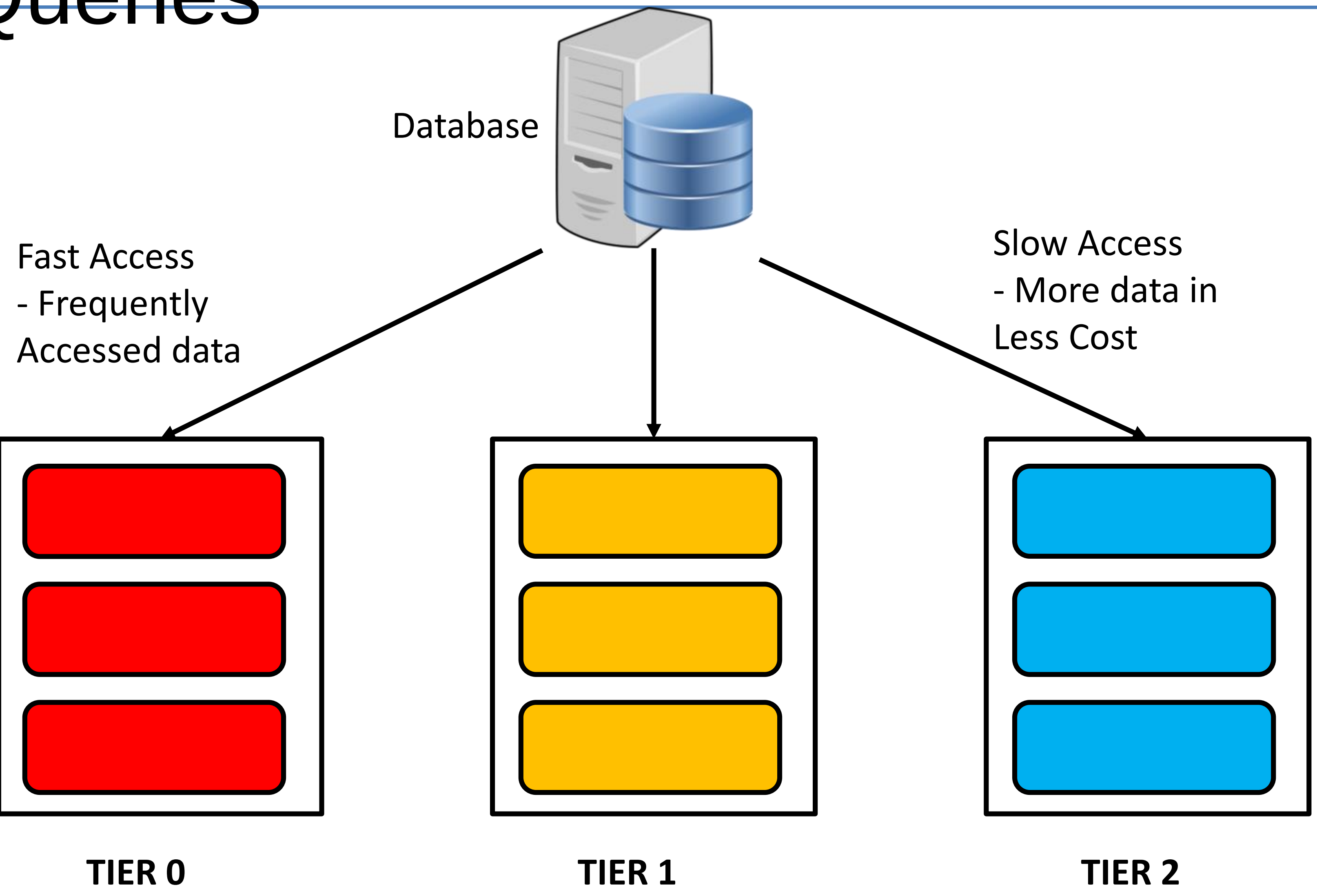


Using ADO in 12c – Database Automates

- Create ADO policies on Segments



Using TIERED Storage- Queries



Lab

- Compression Usage

Summary

- Understanding Hardware Evolution
- Understanding CPU/ Storage Evolution
- What do you ask when you want more Storage
- Using Compression
- Tiered Storage

How to read an AWR Report

Chapter 7

Agenda

- Terms in an AWR Report
- Important Sections of an AWR Report
- Approach to Read an AWR Report
- How to read an AWR in 5 minutes and find the Root Cause

DB Time

- Sum of All execution that happened in the Database
- Sum of All CPU and Wait that happened for all SQL
- Sum of All Database Response Time

Other Terms

- Elapsed Time – Duration of the AWR Begin and End Snaps
- CPU
    - Sockets – No of Processors on Board
    - Cores – No of Cores per Processor
    - CPUs – With Hyperthreading
- Load – Active Running Processes on CPU
- Top Events / Top Foreground Events / Top Wait Events

Interpret an AWR

- Start with the Metadata – Know what you are looking at
- Duration of AWR
- DB Time – Indicates about the Total Wait if present
- Begin and End Snap stats
  - Cursors – Leaking?
  - Sessions – Leaking?
- What are the Top Events
- Logons / Transactions / Parses
- Instance Efficiency
- Time Model Statistics

AWR Report Metadata

- Begin Snapshot
- End Snapshot
- Elapsed Time
- DB Time

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	36367	09-Feb-12 09:00:56	208	16.6
End Snap:	36369	09-Feb-12 11:00:28	223	24.3
Elapsed:		119.53 (mins)		
DB Time:		3,545.81 (mins)		

Host and Instance CPU

- Is the System Loaded Heavily
- Is it used by USER or SYSTEM

Host CPU (CPUs: 16 Cores: 8 Sockets: 4)

Load Average Begin	Load Average End	%User	%System	%WIO	%Idle
32.71	34.88	71.1	28.6	0.0	0.3

Instance CPU

%Total CPU	%Busy CPU	%DB time waiting for CPU (Resource Manager)
92.4	92.8	38.8

Load Profile

- What did the database do during the time of Observation

Load Profile

	Per Second	Per Transaction
Redo size:	93,160.22	4,381.28
Logical reads:	244,241.80	11,486.58
Block changes:	422.54	19.87
Physical reads:	8,142.36	382.93
Physical writes:	45.42	2.14
User calls:	2,032.67	95.60
Parses:	435.25	20.47
Hard parses:	7.42	0.35
Sorts:	132.06	6.21
Logons:	0.16	0.01
Executes:	449.84	21.16
Transactions:	21.26	

Instance Efficiency

- Ratio Analysis of Memory Usage – Target – 100%

Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	100.00	Redo NoWait %:	100.00
Buffer Hit %:	99.94	In-memory Sort %:	100.00
Library Hit %:	86.70	Soft Parse %:	74.12
Execute to Parse %:	62.40	Latch Hit %:	100.00
Parse CPU to Parse Elapsd %:	98.84	% Non-Parse CPU:	82.55

Top Events

- Top Events (Waited)

Top 5 Timed Events

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
CPU time		129,958		61.1	
db file sequential read	20,405,242	36,564	2	17.2	User I/O
gc buffer busy	7,839,623	18,471	2	8.7	Cluster
db file scattered read	3,766,548	13,338	4	6.3	User I/O
gc cr grant 2-way	7,555,461	9,143	1	4.3	Cluster

Time Model Statistics

Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	204,730.73	96.23
DB CPU	129,957.53	61.08
parse time elapsed	5,624.71	2.64
hard parse elapsed time	3,200.25	1.50
PL/SQL execution elapsed time	174.00	0.08
hard parse (sharing criteria) elapsed time	40.00	0.02
hard parse (bind mismatch) elapsed time	35.39	0.02
PL/SQL compilation elapsed time	19.33	0.01
connection management call elapsed time	3.56	0.00
repeated bind elapsed time	2.60	0.00
sequence load elapsed time	1.60	0.00
failed parse elapsed time	1.23	0.00
DB time	212,748.88	
background elapsed time	9,606.63	
background cpu time	9,166.80	

Wait Class

Wait Class

- s - second
- cs - centisecond - 100th of a second
- ms - millisecond - 1000th of a second
- us - microsecond - 1000000th of a second
- ordered by wait time desc, waits desc

Wait Class	Waits	%Time -outs	Total Wait Time (s)	Avg wait (ms)	Waits /txn
User I/O	26,378,885	0.00	54,350	2	172.98
Cluster	23,135,027	0.00	39,015	2	151.70
Commit	145,421	0.00	2,318	16	0.95
Other	631,263	56.49	359	1	4.14
System I/O	283,002	0.00	301	1	1.86
Network	14,259,269	0.00	254	0	93.50
Concurrency	207,494	0.30	167	1	1.36
Application	3,573	1.40	29	8	0.02
Configuration	642	0.16	3	5	0.00

Service Statistics

Service Statistics

- ordered by DB Time

Service Name	DB Time (s)	DB CPU (s)	Physical Reads	Logical Reads
SMPDB01	212,530.50	129,747.50	58,401,323	1,750,797,971
SYS$USERS	296.50	206.40	15,014	240,945
SMPDB01XDB	0.00	0.00	0	0
SYS$BACKGROUND	0.00	0.00	19,519	676,672

SQL Statistics

SQL Statistics

- SQL ordered by Elapsed Time
- SQL ordered by CPU Time
- SQL ordered by Gets
- SQL ordered by Reads
- SQL ordered by Executions
- SQL ordered by Parse Calls
- SQL ordered by Sharable Memory
- SQL ordered by Version Count
- SQL ordered by Cluster Wait Time
- Complete List of SQL Text

SQLs ordered by Elapsed Time

TISYA

SQL ordered by Elapsed Time

- Resources reported for PL/SQL code includes the resources used by all SQL statements called by the code.
- % Total DB Time is the Elapsed Time of the SQL statement divided into the Total Database Time multiplied by 100
- %Total - Elapsed Time as a percentage of Total DB time
- %CPU - CPU Time as a percentage of Elapsed Time
- %IO - User I/O Time as a percentage of Elapsed Time
- Captured SQL account for 90.5% of Total DB Time (s): 3,514,498
- Captured PL/SQL account for 0.7% of Total DB Time (s): 3,514,498

Elapsed Time (s)	Executions	Elapsed Time per Exec (s)	%Total	%CPU	%IO	SQL Id	SQL Module	SQL Text
371,743.24	19,866	18.71	10.58	24.25	0.00	6s9gnzqj59zvz		SELECT [redacted]
149,551.09	19,852	7.53	4.26	25.30	0.00	185zqwthng7ur		SELECT [redacted]
143,251.48	21,841	6.56	4.08	25.16	0.00	7zu4d0vv52c8s		SELECT [redacted]
126,018.92	18,654	6.76	3.59	25.21	0.00	2ty9gy17p6hyz		SELECT [redacted]
122,357.56	18,623	6.57	3.48	26.13	0.00	2rtw43uunsqdm		SELECT [redacted]
111,626.87	0		3.18	0.33	0.01	a2rpxxu6mx1x5		select A[redacted]
109,685.67	26,063	4.21	3.12	0.06	0.00	8u0agzdzfps4n		SELECT [redacted]
109,415.13	26,063	4.20	3.11	30.98	0.00	ccmrxfry60gn1	[redacted])	SELECT [redacted]
106,053.60	18,656	5.68	3.02	25.06	0.00	17bjcwbjtxa8c		SELECT [redacted]
102,047.42	19,177	5.32	2.90	25.39	0.00	gztkaadh0vpa6		SELECT [redacted]
100,945.18	18,644	5.41	2.87	25.29	0.00	1nzdr121cn9vn		SELECT [redacted]
99,021.87	18,672	5.30	2.82	24.93	0.00	7f1ny1jjcwk77		SELECT [redacted]
98,520.16	19,161	5.14	2.80	25.70	0.00	crr1wu2kd8n0a		SELECT [redacted]
98,340.08	19,177	5.13	2.80	25.30	0.00	4baz5a6uyhdck		SELECT [redacted]
96,951.65	19,168	5.06	2.76	25.41	0.00	0uxsi3xfq1cz8		SELECT [redacted]

IO Statistics

IO Stats

- IOStat by Function summary
- IOStat by Filetype summary
- IOStat by Function/Filetype summary
- Tablespace IO Stats
- File IO Stats

What about the Remaining Sections in AWR

- AWR has a lot of Data
- You don’t need to read through everything everytime
- Based on need go to other Sections
- The Pointers will be from the sections discussed

DEMO

- Do you have any AWR to Interpret?

- https://apexapps.oracle.com/pls/apex/f?p=44785:24:15232896062786:PRODUCT:::P24_CONTENT_ID,P24_PREV_PAGE,P24_PROD_SECTION_GRP_ID:10225,141,1745

- https://youtu.be/2QggbUdNsfI

Summary

- Terms in AWR
- Important Sections of an AWR Report
- Approach to Read an AWR Report
- How to read an AWR in 5 minutes and find the Root Cause

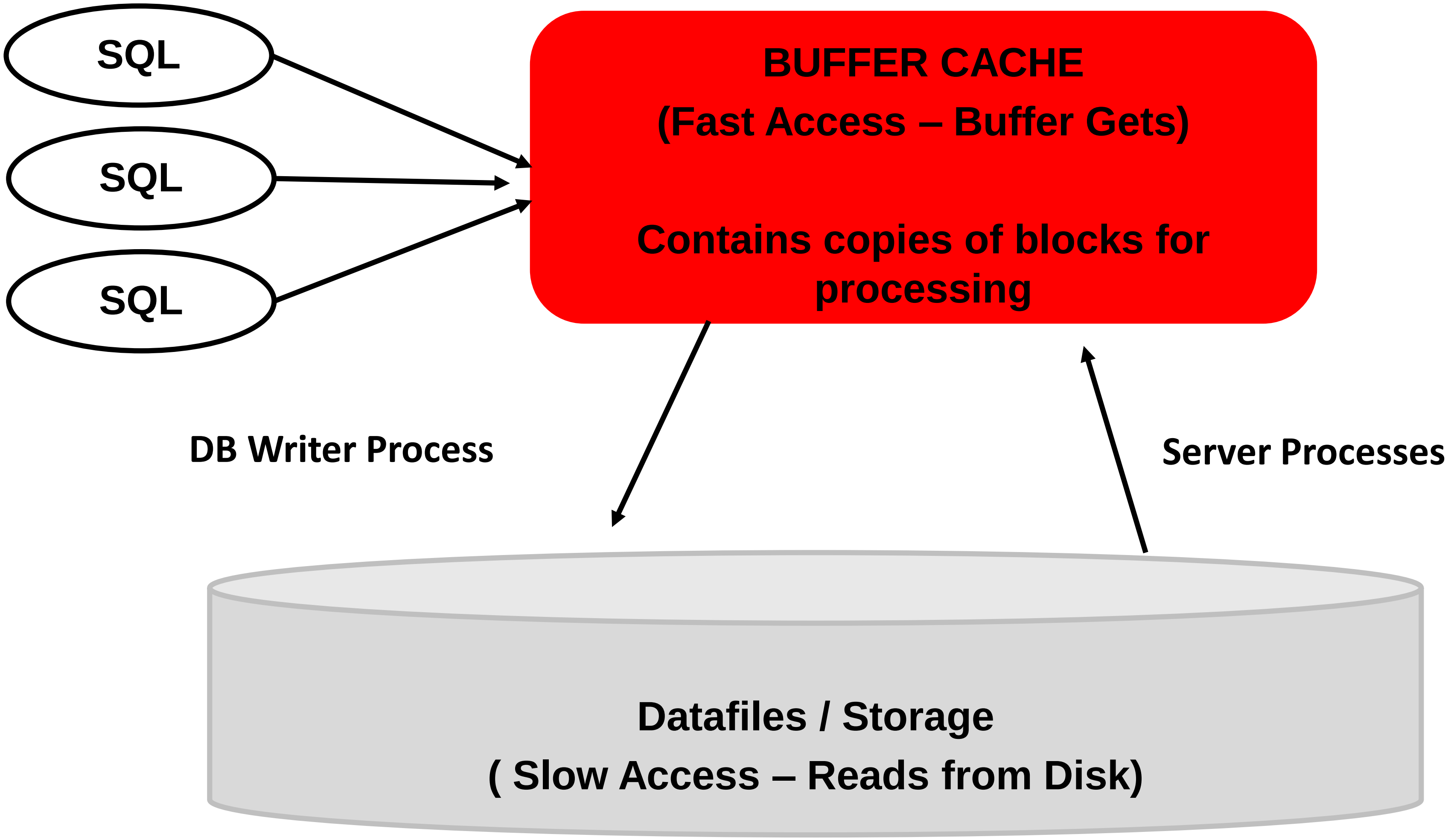
Get the Best out of SGA and Memory

Chapter 8

Agenda

- Optimizing Buffer Cache
- Using Flash Cache
- Tuning Shared Pool
- Sharing Cursors
- Using Result Cache
- Tuning PGA
- FULL Database Caching

Buffer Cache



Optimizing Buffer Cache

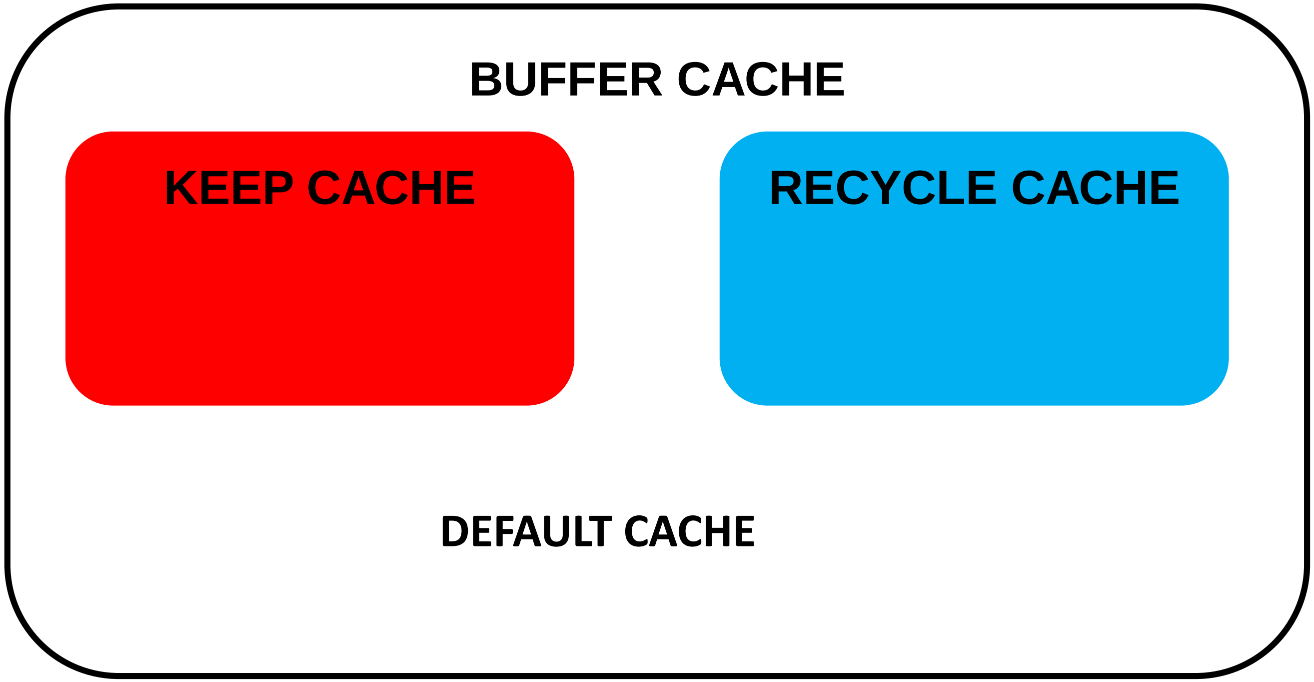
- Size it correctly
- Are you using your Server Memory to the best
- Is your Buffer Hit Ratio100%?
- Do you see lot of
  - Read Waits
  - Free Buffer Waits

- If you have a very large Buffer Cache that can also create Waits to search the Buffer Chain
- Check the Buffer Cache Advisor – MOS - 754639.1

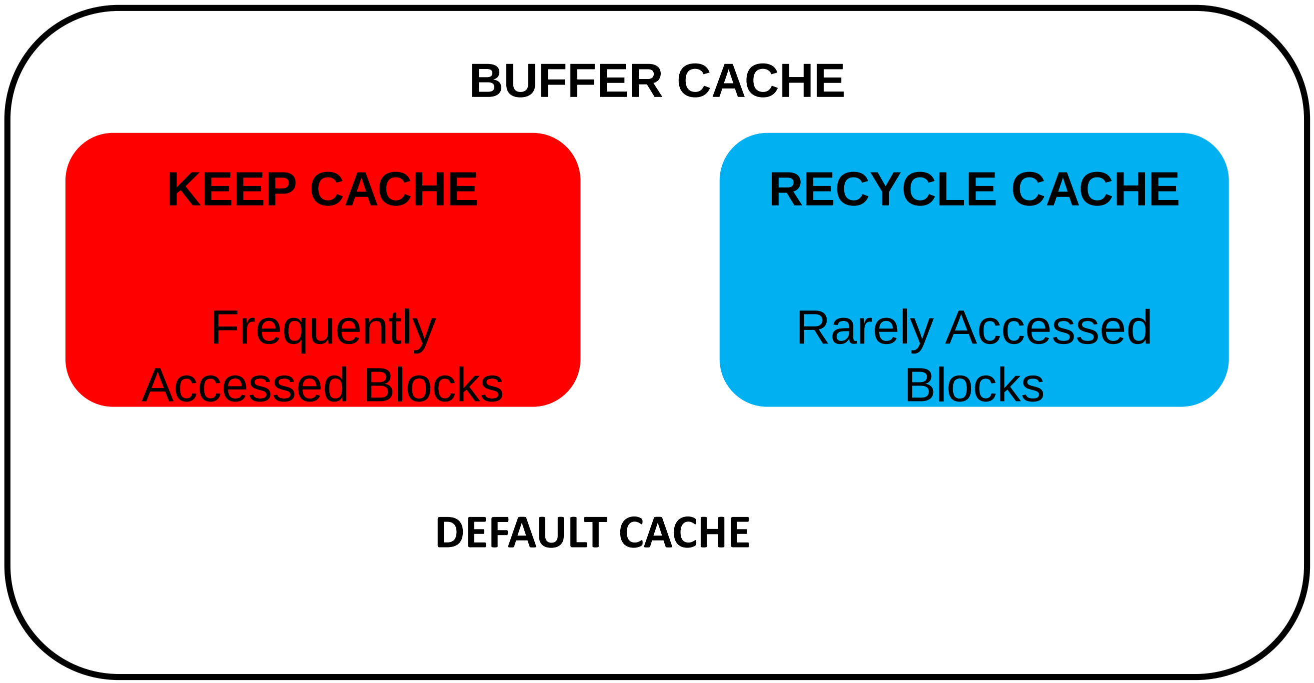
Buffer Cache – Related Waits

- Buffer Busy Waits – Usually due to Hot Blocks
  - Generally Changes in buffer – 1 ms
  - Reading from Disk - 20 ms
- Free Buffer Waits – Slow DBWR or Small Cache
- Cache Buffer chain Latch – Hot Blocks
  - A chain is a set of blocks based on Hash Value
- Cache Buffers LRU chain latch - Use Multiple Pools

Keep / Recycle Cache

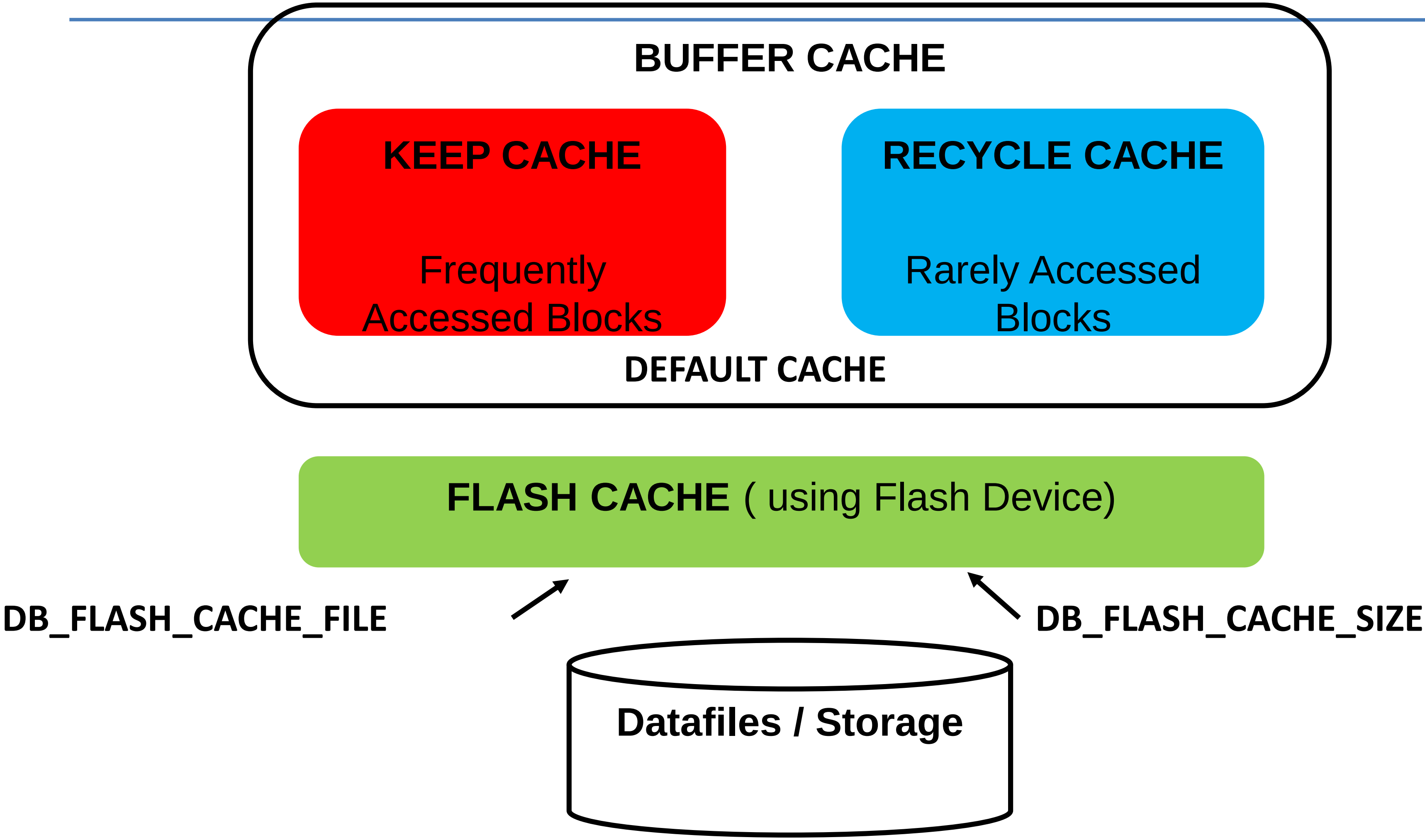


Keep / Recycle Cache



ALTER TABLE <TABLE_NAME> BUFFER_POOL KEEP/RECYCLE

Use Flash Cache



ALTER TABLE <TABLE_NAME> STORAGE (FLASH_CACHE KEEP/NONE/DEFAULT)

Using Flash Cache

- Flash Cash is an additional Storage to be Configured
- Objective is to
  - Use the Faster Storage to Cache Data
  - Blocks flushed from Buffer Cache is stored here
  - Next Access to that Block is faster compared to DISK IO
- Requirements
  - Solaris or Oracle Linux as O/S
  - Buffer Pool Advisory indicates to Double the Size
  - DB File Sequential Read is a top Wait
  - Spare CPU available in Server (Not fully utilized)
- Flash Cache Size – 2 to 10 times of Buffer Cache Size
- Static Parameters – Changes need Restart

Using Flash Cache

- For Each Block into Flash Cache 100bytes / 200 bytes(RAC) is required in Buffer Cache(maintain pointer)
- 1 Flash Cache Device in 11g, Upto 16 in 12c
- A Flash Cache File can be used by 1 Instance Only
- Can configure multiple LUNs from a device and Share it
  - Be Sure about the IO Requirements
- Tables can be configured as
  - NONE
  - KEEP
  - DEFAULT

ALTER TABLE <NAME> STORAGE (FLASH_CACHE <DEFAULT>)

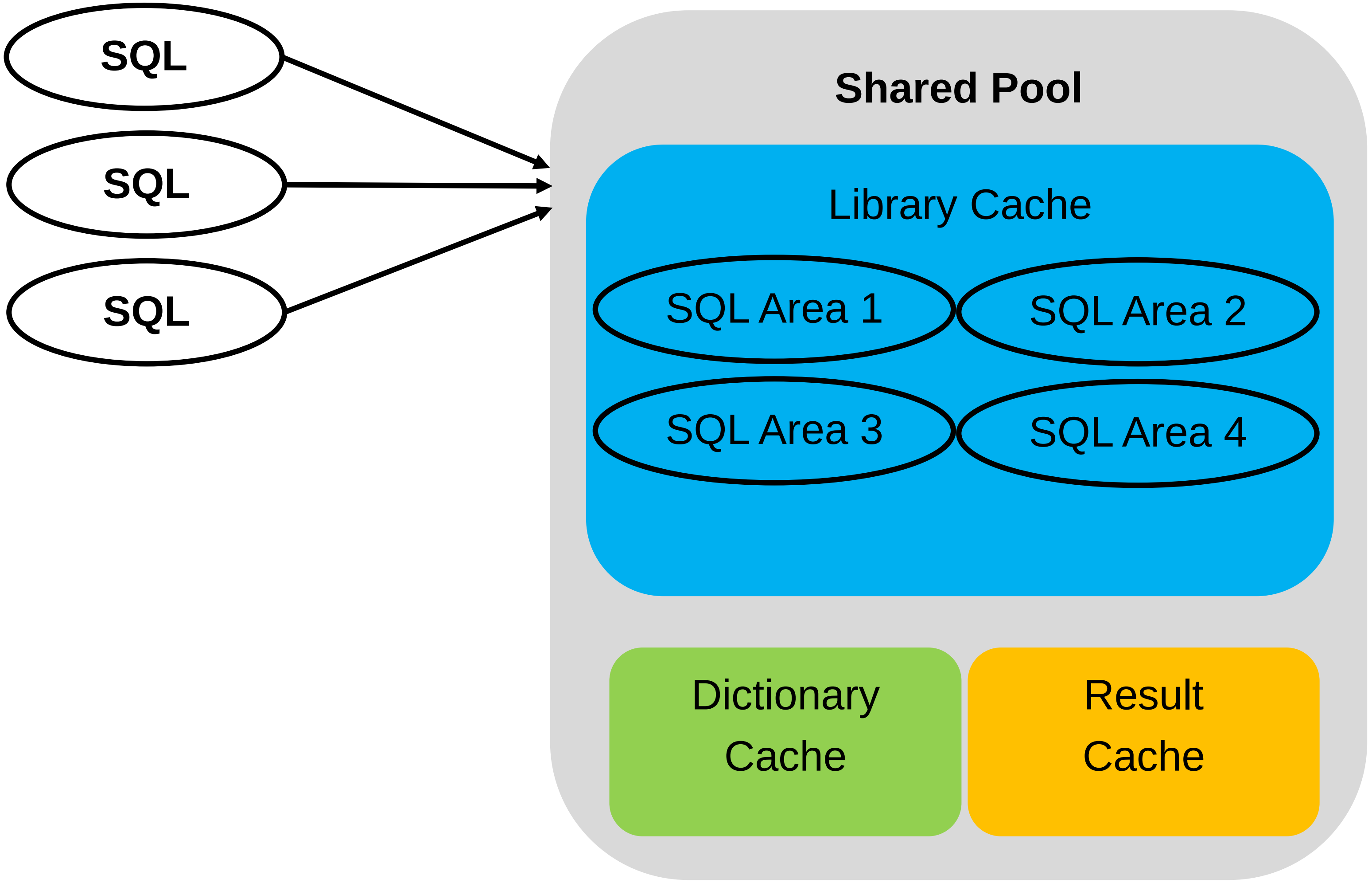
Flash Cache Usage Statistics

- V$FLASHFILESTAT – Cumulative Statistics
  - Singleblkrds – Total Reads
  - Singleblkrdtim_micro – Total Time to do the reads
  - Given for every Flash File
- Optimized Reads
  - UnOptimized Reads – Not from FlashCache

Segments by Optimized Reads

Segments by UnOptimized Reads

Shared Pool


SQL
SQL
SQL
Shared Pool
Library Cache
SQL Area 1
SQL Area 2
SQL Area 3
SQL Area 4
Dictionary Cache
Result Cache

Shared Pool Waits

- Results on account of Exclusive Access to modify Memory Areas
- Latches and Mutexes are acquired to do the same
- Mutexes are more granular hence better **Concurrency**
- Serialization Required to avoid an Object being
  - Deallocated while someone is using it
  - Read when someone is Modifying it
  - Modified when someone is modifying /reading it

    - Read when someone is Reading it… **No Serialization**

Shared Pool Waits

- Cursor : Pin S – Concurrent Read by Many
- Cursor : Pin X – Wait to Get Exclusive Access
- Cursor : Pin S for X – Waiting for someone to Modify
- Cursor : Mutex S – When Too many Child Cursors
- Cursor : Mutex X – Building a New Child for a Parent
- Library Cache : mutex X – Many Reasons
  - Too many Hard Parses
  - High Version Count
  - Invalidations / Reloads
  - Logon / Logoff Storm
- Library Cache : Mutex S

Why Share Cursors

- Scalable Database Requests
- Efficient Usage of Shared Pool
- Use CPU for other purposes
- Reduce Waits in Shared Pool

How to Know if Cursors are Shared

```
SELECT substr(sql_text,1,40) "SQL", count(*) ,
           sum(executions) "TotExecs“
  FROM v$sqlarea
  WHERE executions < 5
  GROUP BY substr(sql_text,1,40)
  HAVING count(*) > 30
  ORDER BY 2;
```

- SELECT plan_hash_value, count(*)
- FROM v$sql
- WHERE parsing_schema_name not in('SYS','SYSMAN','DBSNMP')
- GROUP BY plan_hash_value ORDER BY 2;

What are Child Cursors

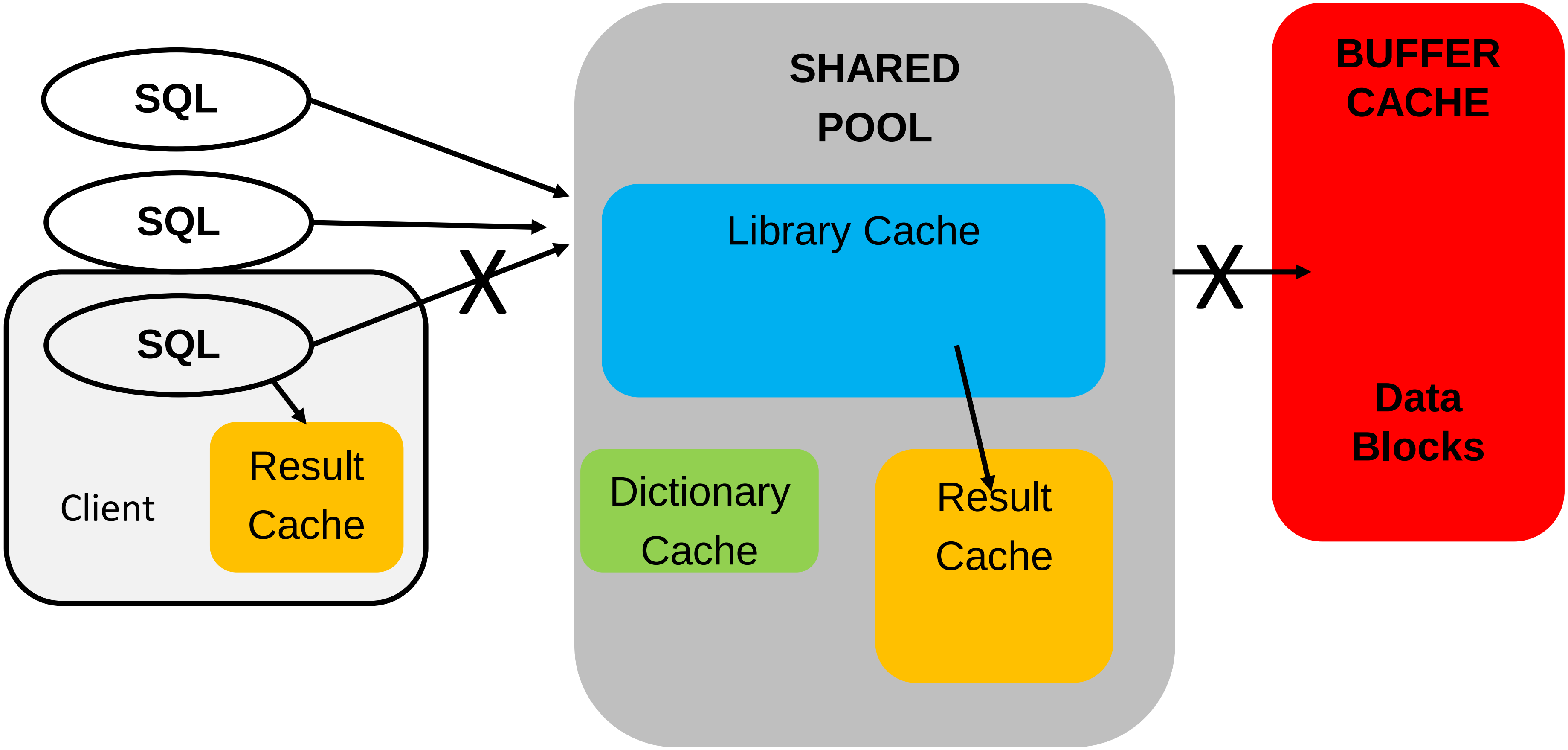
- Parent Cursor – Same SQL Text
- Child Cursor – Different Cursor Created for Same Parent

  Different Metadata for Execution
  - Different Optimizer Environment
  - Executed in a Different Schema, refers different object
  - Flashback
  - Reoptimization feedback
  - Many more – V$SQL_SHARED_CURSOR

Keeping Objects in Shared Pool

- Keep Large PL/SQL Objects in Shared Pool
- Keep Shared Cursors so that they are not thrown out
- DBMS_SHARED_POOL.KEEP
  - SQL_ADDRESS
  - HASH_VALUE

RESULT CACHE



Using Result Cache

- Server Side Result Cache
  - SQL
  - PL/SQL Function
    - Relies On
- Client Side Result Cache
  - Uses the Entire Query Result (not for subqueries)
  - No Visit to Database.. If Same Query Executed
- Instance / Session
  - RESULT_CACHE_MODE – Manual or Force
- HINT in SQL
  - HINT - RESULT_CACHE / NO_RESULT_CACHE
- Table Annotation - DEFAULT / FORCE

Using Server Result Cache

- Parameters
  - RESULT_CACHE_MAX_SIZE
  - RESULT_CACHE_MAX_RESULT
  - RESULT_CACHE_MAX_EXPIRATION
- DBMS_RESULT_CACHE
  - To Influence – BYPASS
  - To Manage – Flush / Invalidate / Status
  - To Report - Report
- Know about Usage
  - V$RESULT_CACHE_OBJECTS
  - V$RESULT_CACHE_STATISTICS
  - V$RESULT_CACHE_MEMORY

Using Client Result Cache

- Parameters
  - CLIENT_RESULT_CACHE_SIZE
  - CLIENT_RESULT_CACHE_LAG
  - COMPATIBLE – for VIEWS -11.2
- Client Can override with Parameters at Client
- Know about usage
  - CLIENT_RESULT_CACHE_STAT$

How is PGA used

- To process data before sending result
  - Sort
  - Hash
  - Join
  - Creating Indexes
- SQLs that use
  - Distinct
  - Aggregate Functions
  - ORDER By
  - Group By
  - Joins

Is PGA used Properly

- Optimal – Entire Operation finished in Memory
- One-Pass – Hits Temporary Tablespace Once
- Mutli-Pass – Hits Temporary Tablespace Multiple Times

How to Start to Size PGA

- Always use only 80% of Memory – 20% for OS
- Of the Available 80% for Database
  - OLTP
    - 20% to PGA, 80% to SGA
  - DWH / DSS
    - 50% to PGA, 50% to SGA
- Monitor your database and modify as required

PGA Statistics

- V$SYSSTAT and V$SESSTAT
  - Workarea memory Allocated
  - Workarea Executions – Optimal
  - Workarea Executions – One Pass
  - Workarea Executions – Multipass
- V$PGASTAT

PGA Advisory - Sizing

- 1- Estd PGA Overallocation Count – 0
  - Correctly Sized PGA (will not use more than Target)
- 2- Estd ExtraW/A MB Read/Write.. - Stabilizes
  - Cant Improve Beyond This
- Many times, you may reach 1 before 2???

PGA Memory Advisory

- When using Auto Memory Mgmt, minimally choose a pga_aggregate_target value where Estd PGA Overalloc Count is 0

PGA Target Est (MB)	Size Factr	W/A MB Processed	Estd Extra W/A MB Read/ Written to Disk	Estd PGA Cache Hit %	Estd PGA Overalloc Count	Estd Time
872	0.13	1,515,143.56	1,029,377.67	60.00	0	542,416,828
1,744	0.25	1,515,143.56	1,029,377.67	60.00	0	542,416,828
3,488	0.50	1,515,143.56	1,029,377.67	60.00	0	542,416,828
5,232	0.75	1,515,143.56	1,029,377.67	60.00	0	542,416,828
6,976	1.00	1,515,143.56	993,339.84	60.00	0	534,734,627
8,371	1.20	1,515,143.56	993,339.84	60.00	0	534,734,627
9,766	1.40	1,515,143.56	993,339.84	60.00	0	534,734,627

Tuning PGA

- Tune SQLs to Avoid using PGA… Is that Easy?
- V$SQL_WORKAREA_HISTOGRAM
  - History of Optimal or Pass executions
- V$SQL_WORKAREA_ACTIVE
  - Current Running Operations
- Know which queries are Using PGA excessively

Serial Reusability

- Use Memory of Variables only for a Call
- Useful for PL/SQL that use Large Memory for Variables
- Public variables don’t maintain state for Session
- Release Memory immediately after a Call
- Does not use PGA, instead uses from SGA

- When Multiple Sessions call the Same Unit

- PRAGMA SERIALLY_REUSABLE in PL/SQL

Free Unused PGA

- Useful to Free Unused PGA in a Session
  - Session Performs an operation requiring Large PGA
  - It can be used again only if a similar operation is performed again
    - SORT area cannot be used for Collection Variable
  - Useful to Ensure Sessions Clear their Memory after performing required Operations

- DBMS_SESSION.FREE_UNUSED_USER_MEMORY

FULL Database Caching (12.1.0.2)

- Nowadays… Servers have a lot of RAM (MEMORY)
- Alter Database force full database Caching; (12.1.0.2)
- This will ensure the Buffer Cache contains the Entire Data
  - Need to estimate Buffer Cache if AMM or ASMM is used
- Data is Cached on Access of Objects
- Even NOCACHE objects are kept in memory

- Applicable at a CDB level only

- Select FORCE_FULL_DB_CACHING from v$database

Lab

– Configuring Multiple Buffer Pools
– Using Result Cache

Summary

- Optimizing Buffer Cache
- Using Flash Cache
- Tuning Shared Pool
- Sharing Cursors
- Using Result Cache
- Tuning PGA
- FULL Database Caching

Identify I/O Hotspots

Chapter 9

Agenda

- Getting I/O statistics from
  - Dynamic Performance Views
  - AWR Report
- File and I/O Statistics
- Object I/O Statistics

Expected Thresholds for I/O

- 64 Blocks of 8k(512Kb) – 20 ms
- Smaller IO Requests – 10-20ms
- Single Block Operations faster than Multi-Block
- <=15 ms for
  - Log File parallel Write
  - Control File Write
  - Direct Path Writes

- MOS - 1275596.1

Views giving this Information

- SELECT sid, total_waits, time_waited FROM v$session_event WHERE event='db file sequential read' and total_waits>0 ORDER BY 3,2 ;

- V$SQL – DISK_READS
- V$SESSTAT- Physical Reads

Is I/O a Problem

- TOP Timed Events in AWR is starting point

Top 5 Timed Events

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
db file sequential read	2,053,765	6,596	3	55.2	User I/O
CPU time		3,628		30.4	
db file scattered read	197,123	781	4	6.5	User I/O
SQL*Net more data from dblink	51,722	523	10	4.4	Network
log file sync	24,037	296	12	2.5	Commit

10g

Top 5 Timed Foreground Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
db file scattered read	267,477	13,203	49	28.44	User I/O
db file sequential read	1,876,980	9,595	5	20.67	User I/O
DB CPU		8,179		17.62	
log file sync	240,065	5,478	23	11.80	Commit
library cache: mutex X	3,362	2,378	707	5.12	Concurrency

11g R2

Top 10 Foreground Events by Total Wait Time

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time	Wait Class
db file scattered read	3,851,476	6982	1.81	58.0	User I/O
DB CPU		3626,8		30.1	
db file sequential read	875,861	607,2	0.69	5.0	User I/O
direct path write temp	308,249	589,9	1.91	4.9	User I/O
direct path read temp	134,700	205,5	1.53	1.7	User I/O
read by other session	112,627	83	0.74	.7	User I/O
direct path read	77,511	43,4	0.56	.4	User I/O
control file sequential read	55,660	18,6	0.33	.2	System I/O
log file sync	9,220	16,1	1.75	.1	Commit
enq: FB - contention	1,355	8,4	6.24	.1	Other

12c

AWR I/O Stats

10g

IO Stats

- Tablespace IO Stats
- File IO Stats

11g Onwards

IO Stats

- IOStat by Function summary
- IOStat by Filetype summary
- IOStat by Function/Filetype summary
- Tablespace IO Stats
- File IO Stats

AWR I/O Stats – Function Summary

IOStat by Function summary

- 'Data' columns suffixed with M,G,T,P are in multiples of 1024 other columns suffixed with K,M,G,T,P are in multiples of 1000
- ordered by (Data Read + Write) desc

Function Name	Reads: Data	Reqs per sec	Data per sec	Writes: Data	Reqs per sec	Data per sec	Waits: Count	Avg Tm(ms)
Buffer Cache Reads	879,3G	262.60	49,984M	0M	0.00	0M	4,7M	1.61
Direct Reads	120,1G	50.67	6,83M	4G	0.97	,227M	0	
Direct Writes	9,7G	12.16	,551M	105,9G	26.47	6,02M	0	
Others	3,7G	5.73	,213M	1,2G	1.33	,069M	112,2K	0.45
RMAN	3,1G	0.28	,175M	137M	0.04	,008M	4593	1.51
LGWR	0M	0.00	0M	605M	2.74	,034M	24,6K	1.47
DBWR	0M	0.00	0M	416M	0.94	,023M	0	
TOTAL:	1015,9G	331.44	57,752M	112,2G	32.48	6,379M	4,9M	1.58

AWR I/O Stats – File Type Summary

IOStat by Filetype summary

- 'Data' columns suffixed with M,G,T,P are in multiples of 1024 other columns suffixed with K,M,G,T,P are in multiples of 1000
- Small Read and Large Read are average service times, in milliseconds
- Ordered by (Data Read + Write) desc

Filetype Name	Reads: Data	Reqs per sec	Data per sec	Writes: Data	Reqs per sec	Data per sec	Small Read	Large Read
Data File	899,2G	267.27	51,119M	421M	0.96	,023M	0.24	1.31
Temp File	109,9G	58.20	6,247M	109,9G	27.45	6,247M	1.48	3.77
Control File	5,8G	5.90	,33M	720M	1.21	,04M	0.02	2.71
Archive Log	683M	0.04	,038M	340M	0.02	,019M	0.00	121.46
Log File	340M	0.02	,019M	603M	2.74	,033M	0.02	6.26
Flashback Log	2M	0.00	0M	300M	0.11	,017M	0.22	
Other	0M	0.00	0M	0M	0.00	0M	0.17	
TOTAL:	1015,9G	331.44	57,752M	112,2G	32.48	6,379M	0.66	1.54

AWR I/O Stats - Tablespace

Tablespace IO Stats

- ordered by IOs (Reads + Writes) desc

Tablespace	Reads	Av Reads/s	Av Rd(ms)	Av Blks/Rd	Writes	Av Writes/s	Buffer Waits	Av Buf Wt(ms)
	550,035	77	0.54	1.00	7,590	1	0	0.00
	328,915	46	0.32	1.00	267	0	0	0.00
	241,432	34	6.36	1.00	43,910	6	0	0.00
	217,063	30	3.64	1.62	126	0	33,918	3.42
	153,122	21	4.79	1.01	7,813	1	0	0.00
	117,027	16	9.96	1.00	26,620	4	0	0.00
	103,171	14	2.47	1.00	48	0	273	1.61
	66,655	9	3.63	1.00	14,274	2	0	0.00
	68,203	9	4.03	6.57	332	0	6	5.00

AWR I/O Stats – File I/O

File IO Stats

- ordered by Tablespace, File

Tablespace	Filename	Reads	Av Rds/s	Av Rd(ms)	Av Blks/Rd	1-bk Rds/s	Av 1-bk Rd(ms)	% Opt Reads	Writes	Writes avg/s	Buffer Waits	Av Buf Wt(ms)
[redacted]	[redacted]	3	0	0.00	1.00	0	0.00	0.00	3	0	0	0.00
[redacted]	[redacted]	3	0	3.33	1.00	0	3.33	0.00	3	0	0	0.00
[redacted]	[redacted]	6	0	1.67	1.00	0	1.67	0.00	3	0	0	0.00
[redacted]	[redacted]	6	0	1.67	1.00	0	1.67	0.00	3	0	0	0.00
[redacted]	[redacted]	14,443	1	0.80	1.00	1	0.80	0.00	437	0	12,494	0.57
[redacted]	[redacted]	1,228	0	0.94	1.00	0	0.95	0.00	28	0	156	0.58
[redacted]	[redacted]	1,330	0	0.92	1.00	0	0.92	0.00	7	0	178	0.73
[redacted]	[redacted]	87,071	5	0.79	1.27	5	0.75	0.00	2,285	0	101,651	0.73
[redacted]	[redacted]	2,823	0	0.84	1.16	0	0.80	0.00	3	0	1,055	1.01
[redacted]	[redacted]	2,914	0	0.81	1.17	0	0.79	0.00	285	0	1,088	0.68
[redacted]	[redacted]	9	0	0.00	1.00	0	0.00	0.00	9	0	0	0.00
[redacted]	[redacted]	41,826	2	1.24	15.90	0	1.67	0.00	6	0	0	0.00
[redacted]	[redacted]	40,323	2	1.05	15.90	0	3.33	0.00	3	0	0	0.00
[redacted]	[redacted]	3	0	0.00	1.00	0	3.33	0.00	3	0	0	0.00
[redacted]	[redacted]	22,909	1	0.94	9.54	1	0.57	0.00	4,689	0	1,360	0.23

AWR – Top SQL

- TOP SQL by Reads

SQL ordered by Elapsed Time

Elapsed Time (s)	Executions	Elapsed Time per Exec (s)	%Total	%CPU	%IO	SQL Id	SQL Module	SQL Text
53,536.36	28	1,912.01	48.89	1.02	10.52			
12,957.44	30	431.91	11.83	1.28	34.43			
8,444.33	35	241.27	7.71	0.01	94.72			
8,427.77	34	247.88	7.70	0.01	94.87			
6,940.85	27	257.07	6.34	0.00	93.04			
1,599.62	16	99.98	1.46	0.02	90.19			
1,111.06	5	222.21	1.01	0.01	99.82		JDBC Thin Client	

SQL ordered by Gets

Buffer Gets	Executions	Gets per Exec	%Total	Elapsed Time (s)	%CPU	%IO	SQL Id	SQL Module	SQL Text
34,694,289	38	913,007.61	35.58	240.34	97.08	0.65			...
33,107,387	28	1,182,406.68	33.95	53,536.36	1.02	10.52			
22,274,208	456	48,846.95	22.84	129.81	100.33	0.28			
8,038,911	3,572	2,250.53	8.24	35.51	101.72	0.12			
3,219,572	49	65,705.55	3.30	157.08	99.98	0.01			
2,632,525	30	87,750.83	2.70	12,957.44	1.28	34.43			
2,342,422	6,286	372.64	2.40	95.09	100.21	0.13		JDBC Thin Client	

AWR – Segment Statistics

Segment Statistics

- Segments by Logical Reads
- Segments by Physical Reads
- Segments by Physical Read Requests
- Segments by UnOptimized Reads
- Segments by Optimized Reads
- Segments by Direct Physical Reads
- Segments by Physical Writes
- Segments by Physical Write Requests
- Segments by Direct Physical Writes
- Segments by Table Scans
- Segments by DB Blocks Changes
- Segments by Row Lock Waits
- Segments by ITL Waits
- Segments by Buffer Busy Waits

Consider Tuning SQL

- Can we select only specific rows (PREDICATE)
- Are there Selective Indexes
- Are you using Best Access Structures
  – Alternate Storage Techniques
- Can Data be cached – Multiple Buffer Pools

Lab

- How to reduce I/O

Identify Bad SQL

Chapter 10

Agenda

- What is Bad SQL
- What is Bad I/O
- Are there Thresholds for Bad SQL
- Single Execution or Multiple Executions
- What next after Identifying Bad SQL

What is Bad SQL

- Response time too Long
- Too many Waits
- Consumes lots of Resources (CPU / IO / Memory)

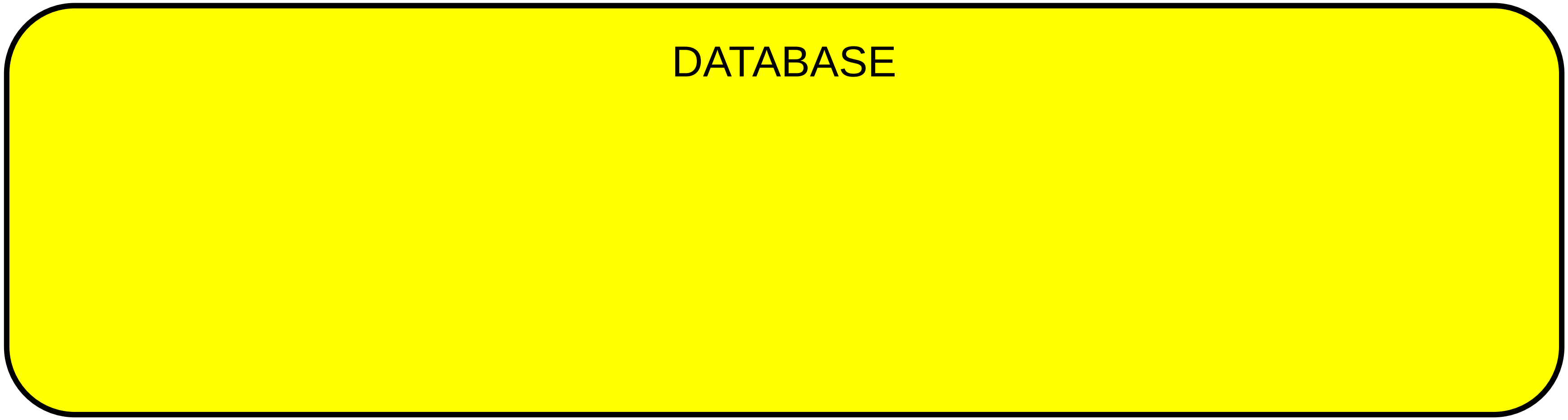
- Bottom Line… Affects SLA of itself or impacts other SQLs

What is Bad I/O

- Is there something called Bad I/O
- Doing Excessive I/O is Bad, try to reduce
- If an SQL Does Disk Reads is it Bad?
- Does an SQL Have control on where its data will be?

Is it Good to do
More Disk Reads or More Memory Reads

- Can SGA hold all data in the Database?
- Will an SQL know if its data is in the SGA?



Is it Good to do More Disk Reads or More Memory Reads

- The Sum of Buffer Gets + Disk Reads is the total Blocks required for I/O
- Always look at total Reads when looking at SQL Performance
- Wrong to decide an SQL is bad because its doing Disk Reads
- Cache table data in Buffer Cache so that SQLs find the data in Memory/ SGA

Are there Thresholds for Bad SQL

- Every drop counts – to Save Water
- If you want better Throughput save
  - Every Second of CPU
  - Every KB of Memory
  - Every IO Operation
- Then your database is scal
- Remember the Bowl

Single Execution or Multiple Executions

- Some queries tear down your Database Because
  - They execute once and use a lot of Resources
  - They Execute a million times.. Though for each execution resource consumption is low
- How to approach this?
- Top SQLs in AWR?

Identify the Problem SQL

- Is it a problem with SQL – Join Missing
- Is the query affecting other Queries?
- Look at reducing the Wait or Resource consumed
  - CPU / MEMORY / IO
  - WAITS?
- Where to get the Information
  - ASH
  - AWR report
  - SQL Monitoring Report
  - Performance Views

What next… After Identifying the Bad SQL

Tune or Increase Resources

- Increase Resources
  - Consumer group Mappings
  - Instance Caging – CPU to instance
  - If Exadata – IORM
  - Get Faster Hardware (CPU / Memory / IO)
- Tune
  - Check SQL Logic
  - Indexes Missing?
  - Alter the Storage Structure
  - Reorganization required?
  - Optimizer Parameters
  - Hints

Summary

- What is Bad SQL
- What is Bad I/O
- Are there Thresholds for Bad SQL
- Single Execution or Multiple Executions
- What next after Identifying Bad SQL

Appendix of References

MOS Notes – AWR / ADDM

- Automatic Workload Repository (AWR) Reports - Main Information Sources - 1363422.1
- AWR Report Interpretation Checklist for Diagnosing Database Performance Issues - 1628089.1
- How to Use AWR Reports to Diagnose Database Performance Issues - 1359094.1
- How to Read PGA Memory Advisory Section in AWR and Statspack Reports - 786554.1
- How to Read Buffer Cache Advisory Section in AWR and Statspack Reports- 754639.1
- How to Interpret the "SQL ordered by Physical Reads (UnOptimized)" Section in AWR Reports (11.2 onwards) For Smart Flash Cache Database- 1466035.1
- How to Interpret the OS stats section of an AWR report - 762526.1

MOS Notes – Buffer Cache

- Understanding and Tuning Buffer Cache and DBWR - 62172.1
- How to Read Buffer Cache Advisory Section in AWR and Statspack Reports - 754639.1
- Oracle Multiple Buffer Pools Feature - 135223.1
- WAITEVENT: "buffer busy waits" Reference Note -34405.1
- WAITEVENT: "latch: cache buffers chains" Reference Note - 2098064.1
- Troubleshooting 'latch: cache buffers chains' Wait Contention -1342917.1
- How to Identify Hot Blocks Within the Database Buffer Cache that may be Associated with 'latch: cache buffers chains' Wait Contention -163424.1
- Resolving Issues Where Waits for 'latch: cache buffers chains' Seen Due to Poorly Tuned SQL - 1476044.1
- Troubleshooting 'latch: cache buffers chains' Wait Contention -1342917.1

MOS Notes – Shared Pool

- Troubleshooting: Understanding and Tuning the Shared Pool - 62143.1
- How to Calculate Your Shared Pool Size - 1012046.6
- Using the Oracle DBMS_SHARED_POOL Package - 61760.1
- FAQ: What are Latches and What Causes Latch Contention (11g and Above) - 1970450.1
- VIEW: "V$SQL_SHARED_CURSOR" Reference Note - 120655.1
- Troubleshooting: Waits for Mutex Type Events -1377998.1
- Troubleshooting 'library cache: mutex X' Waits -1357946.1
- Troubleshooting 'cursor: pin S wait on X' waits - 1349387.1
- FAQ: 'cursor: mutex ..' / 'cursor: pin ..' / 'library cache: mutex ..' Type Wait Events - 1356828.1

MOS Notes – Library Cache

- How to Identify Which Latch is Associated with a "latch free" wait (Post-11g) - 413942.1
- FAQ: What are Latches and What Causes Latch Contention (Pre-11g) - 22908.1
- Troubleshooting: High Version Count Issues -296377.1
- High SQL Version Counts - Script to determine reason(s) - 438755.1
- WAITEVENT: "cursor: pin S" Reference Note -1310764.1
- WAITEVENT: "library cache pin" Reference Note -34579.1
- How to Reduce 'LIBRARY CACHE LATCH' Contention Using a Procedure to KEEP Cursors Executed> 10 times - 130699.1

MOS Notes – Others

- Troubleshooting: High CPU Utilization -164768.1
- Best Practices: Proactive Data Collection for Performance Issues - 1477599.1
- WAITEVENT: "db file sequential read" Reference Note - 34559.1
- How to Tell if the I/O of the Database is Slow - 1275596.1
- Troubleshooting I/O Related Waits -223117.1
- Resolving Issues Where Application Queries are Waiting Too Frequently for 'db file sequential read' Operations - 1475825.1
- Resolving Issues Where Application Queries are Waiting Too Long for 'db file sequential read' Operations Due to Underlying I/O Issues -1477209.1
- WAITEVENT: "log file sync" Reference Note -34592.1
- Troubleshooting: 'Log file sync' Waits-1376916.1

MOS Notes – Others

- Troubleshooting Performance Issues - 1377446.1
- Best Practices: Proactively Avoiding Database and Query Performance Issues - 1482811.1
- OSWatcher (Includes: [Video]) 0 301137.1
- TROUBLESHOOTING: Possible Causes of Poor SQL Performance - 33089.1
- FAQ- Statspack Complete Reference (Doc ID 94224.1)-94224.1
- Automatic Memory Management (AMM) on 11g & 12c - 443746.1
- Automatic PGA Memory Management -223730.1
- How to Read PGA Memory Advisory Section in AWR and Statspack Reports to Tune PGA_AGGREGATE_TARGET - 786554.1
- Archive of Database Performance Related Webcasts and Videos - 1597373.1
- Recommended Method for Obtaining 10046 trace for Tuning - 376442.1

Books and Videos

- Cost-Based Oracle Fundamentals (Expert's Voice in Oracle)
  - Jonathan Lewis
- Effective Oracle by Design (Oracle Press)
  - Thomas Kyte
- Troubleshooting Oracle Performance
  - Christian Antognini
- Oracle Real World Performance – Oracle Learning Library
  - https://apexapps.oracle.com/pls/apex/f?p=44785:141:0::NO::P141_PAGE_ID,P141_SECTION_ID:119,870