ORACLE®

Oracle数据库性能调优

金丹

资深技术顾问

ORACLE 甲骨文

最重要的规则

- 最少的参数，最少的性能影响
- 除非你很清楚你在干什么，否则不要设置 _****
- 除非你很清楚你在干什么，否则不要设置 event

定期监控

- DB statspack,AWR,ADDM
- OS ,vmstat,sar….
- Network
- Grid Control
- RUEI

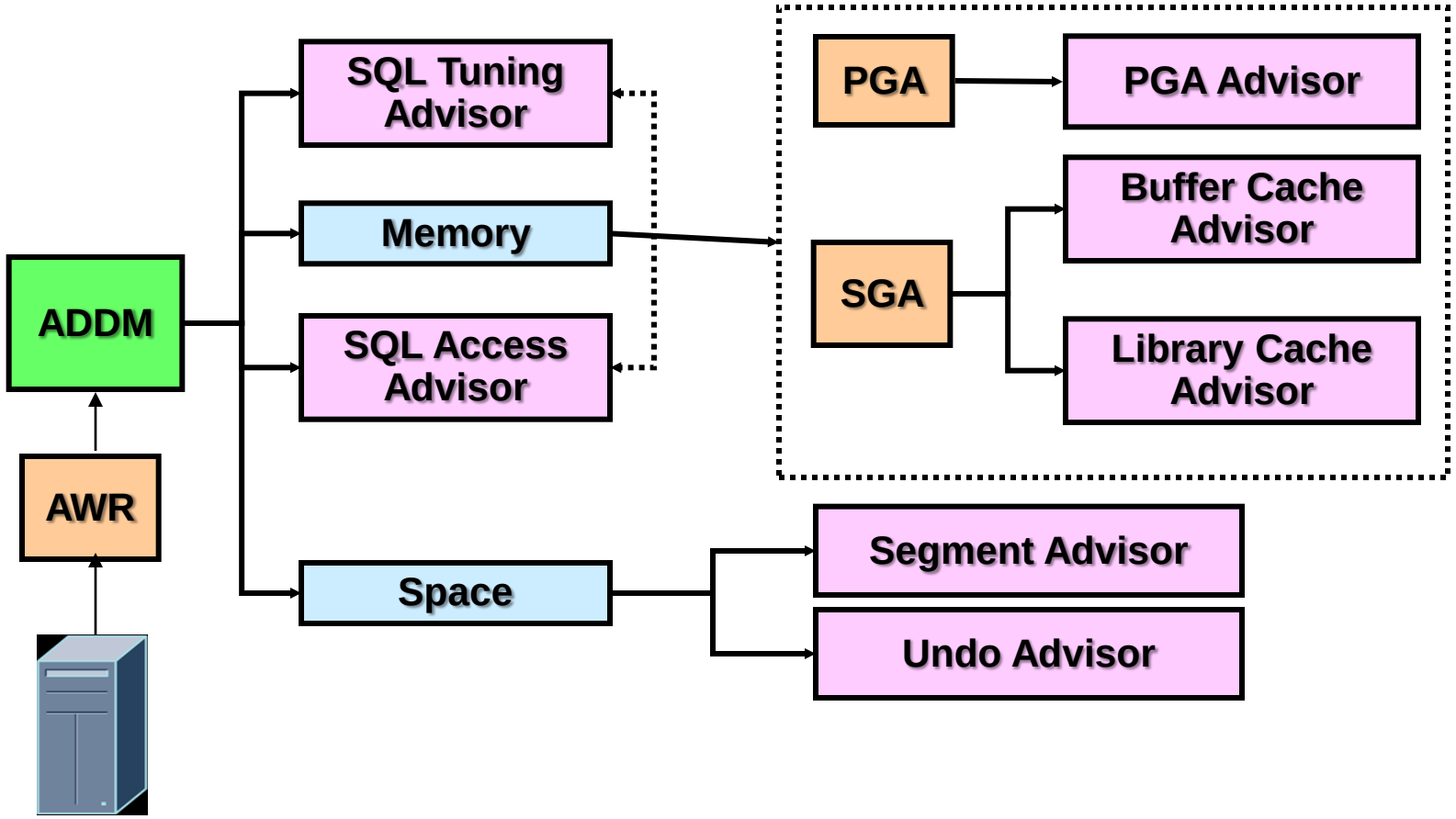
Oracle Tuning Methods

- Prehistoric (v5)
  - Debug code
- Dark Ages (v6)
  - Counters/Ratios
  - BSTAT/ESTAT
  - SQL*Trace
- Renaissance (v7)
  - Introduction of Wait Event instrumentation
  - Move from counters to timers
  - STATSPACK

The Modern Age of Time-based Tuning

- YAPP (8i) - Instance tuning using instance statistics
  - Non intrusive, always on
  - Broadly scoped
- Method R (9i) - Session tuning using 10046 SQL traces
  - Tightly scoped
  - Must be highly selective
- DB Time Tuning (10g) – Comprehensive tuning using fundamental notion of time in database
  - Multiple scoping levels
  - Always on, non-intrusive
  - Built into infrastructure: instrumentation, ASH, AWR, ADDM, EM

Oracle 10g/11g的智能化工具

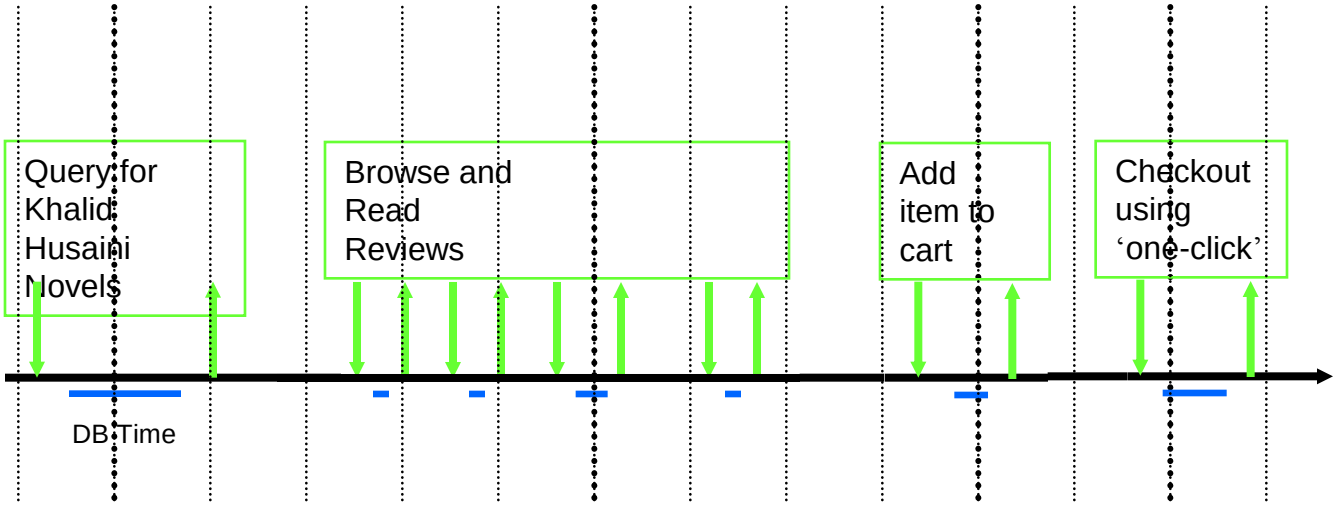


Active Session History (ASH)

- Samples active sessions every one second into memory (v$active_session_history)
- Direct access to kernel structures
- One of ten samples flushed to AWR at every snapshot
- Data captured includes:
  - SID
  - SQL ID
  - Program, Module, Action
  - Wait event#
  - Object, File, Block
  - actual wait time (if captured while waiting)

Enables targeted performance analysis of transient problems

Active Session History (ASH)



Time	SID	Module	SQL ID	State	Event
7:38:26	213	Book by author	qa324jffritcf	WAITING	db file sequential read
7:42:35	213	Get review id	aferv5desfzs5	CPU	
7:50:59	213	Add to cart	hk32pekfcbdfr	WAITING	buffer busy wait
7:52:33	213	One click	abngldf95f4de	WAITING	log file sync

ASH Report: Main Sections

ASH Report For BUGAP/bug1ap

(1 Report Target Specified)

DB Name	DB Id	Instance	Inst num	Release	RAC	Host
BUGAP	1679034986	bug1ap	1	10.2.0.1.0	YES	dbs232

CPUs	SGA Size	Buffer Cache	Shared Pool	ASH Buffer Size
4	2,576M (100%)	1,200M (46.6%)	1,109M (43.0%)	8.0M (0.3%)

	Sample Time	Data Source
Analysis Begin Time:	21-Sep-06 13:13:20	V$ACTIVE_SESSION_HISTORY
Analysis End Time:	21-Sep-06 13:18:20	V$ACTIVE_SESSION_HISTORY
Elapsed Time:	5.0 (mins)	Missing 1.0 mins (20%) of activity
Sample Count:	1,330	
Average Active Sessions:	4.43	
Avg. Active Session per CPU:	1.11	
Report Target:	SQL_ID like 'cyaj7dkrbqs95'	4% of total database activity

ASH Report

- Top Events
- Load Profile
- Top SQL
- Top Sessions
- Top Objects/Files/Latches
- Activity Over Time

ASH Report: Top Events

Top User Events

Event	Event Class	% Activity	Avg Active Sessions
db file sequential read	User I/O	68.80	3.05
gc buffer busy	Cluster	12.33	0.55
buffer busy waits	Concurrency	9.25	0.41
read by other session	User I/O	5.64	0.25
gc cr disk read	Cluster	1.28	0.06

ASH Report: Top SQL

Top SQL Statements

SQL ID	Planhash	% Activity	Event	% Event	SQL Text
cyaj7dkrbqs95	2770286798	97.74	db file sequential read	67.67	select a.product_id, a.t...
			gc buffer busy	12.18	
			buffer busy waits	9.02	

Complete List of SQL Text

SQL Id	SQL Text
cyaj7dkrbqs95	select a.product_id, a.test_name, a.rptno , a.status, to_char(a.rptdate, 'mm/dd/yyyy'), a.base_rptno, b.product_id, a.do_by_release from rpthead a, rpthead b where a.product_id in (501) and a.test_name is not NULL and (a.subject like '%R12.FIN.A.XP%' or a.subject like '%R12.FIN.A.QA%' or a.subject like '%R12.FIN.A.UI%') and a.base_rptno = b.rptno(+) order by a.test_name, a.rptno

ASH Report: Activity Over Time

Activity Over Time

- Analysis period is divided into smaller time slots
- Top 3 events are reported in each of those slots
- 'Slot Count' shows the number of ASH samples in that slot
- 'Event Count' shows the number of ASH samples waiting for that event in that slot
- '% Event' is 'Event Count' over all ASH samples in the analysis period

Slot Time (Duration)	Slot Count	Event	Event Count	% Event
13:14:00 (1.0 min)	220	db file sequential read	163	12.26
		gc buffer busy	27	2.03
		buffer busy waits	12	0.90
13:15:00 (1.0 min)	295	db file sequential read	222	16.69
		gc buffer busy	32	2.41
		buffer busy waits	22	1.65
13:16:00 (1.0 min)	305	db file sequential read	211	15.86
		gc buffer busy	43	3.23
		read by other session	23	1.73
13:17:00 (1.0 min)	295	db file sequential read	199	14.96
		gc buffer busy	35	2.63
		buffer busy waits	28	2.11
13:18:00 (20 secs)	100	db file sequential read	46	3.46
		buffer busy waits	27	2.03
		gc buffer busy	14	1.05

Automatic Workload Repository (AWR)

- Built-in workload and performance statistics repository in the database
- Automatically Captures Workload Data
  - Every 60 minutes, or manually, saves data for 8 days by default
- Resides in SYSAUX tablespace
- Space requirements automatically managed
  - Old data is automatically purged nightly based on retention interval
- Stores different classes of data:
  - BASE STATISTICS e.g., physical reads
  - SQL STATISTICS e.g., disk reads (per sql stmt)
  - METRICS e.g., physical reads / sec
  - **ACTIVE SESSION HISTORY (ASH)**

AWR Compare Period Report

WORKLOAD REPOSITORY COMPARE PERIOD REPORT

Snapshot Set	DB Name	DB Id	Instance	Inst num	Release	Cluster	Host	Std Block Size
First (1st)	PROD	1545480911	prod	1	11.1.0.6.0	NO	tdsrat01-d1.oracleads.com	8192
Second (2nd)	DEV	3707267492	dev	1	11.1.0.6.0	NO	tdsrat01-d2.oracleads.com	8192

Snapshot Set	Begin Snap Id	Begin Snap Time	End Snap Id	End Snap Time	Avg Active Users	Elapsed Time (min)	DB time (min)
1st	339	27-Mar-08 20:17:34 (Thu)	340	27-Mar-08 20:22:07 (Thu)	0.68	4.54	3.09
2nd	396	27-Mar-08 20:52:52 (Thu)	397	27-Mar-08 20:56:18 (Thu)	0.30	3.44	1.04
%Diff					*-55.88*	*-24.23*	*-66.40*

AWR Compare Period Report: Load Profile

Load Profile

	1st per sec	2nd per sec	%Diff	1st per txn	2nd per txn	%Diff
DB time:	0.68	0.30	-55.88	0.09	0.03	-66.67
CPU time:	0.36	0.20	-44.44	0.05	0.02	-60.00
Redo size:	141,784.30	186,369.33	31.45	18,478.25	18,542.89	0.35
Logical reads:	30,539.38	1,289.19	-95.78	3,980.09	128.27	-96.78
Block changes:	726.20	949.25	30.71	94.64	94.45	-0.20
Physical reads:	6,790.88	0.61	-99.99	885.03	0.06	-99.99
Physical writes:	2.88	1.68	-41.67	0.38	0.17	-55.26
User calls:	338.11	447.90	32.47	44.06	44.56	1.13
Parses:	15.58	17.39	11.62	2.03	1.73	-14.78
Hard parses:	0.83	0.24	-71.08	0.11	0.02	-81.82
Sorts:	4.57	9.52	108.32	0.60	0.95	58.33
Logons:	0.09	0.11	22.22	0.01	0.01	0.00
Executes:	344.89	449.90	30.45	44.95	44.76	-0.42
Transactions:	7.67	10.05	31.03			

	1st	2nd	Diff
% Blocks changed per Read:	2.38	73.63	71.25
Recursive Call %:	28.01	21.66	-6.35
Rollback per transaction %:	0.96	0.68	-0.28
Rows per Sort:	51.54	11.15	-40.39
Avg DB time per Call (sec):	0.00	0.00	-0.00

AWR Compare Period Report: Top Timed Events

Top Timed Events

- Events with a "-" did not make the Top list in this set of snapshots, but are displayed for comparison purposes

1st						2nd					
Event	Wait Class	Waits	Time (s)	Avg Time (ms)	%DB time	Event	Wait Class	Waits	Time (s)	Avg Time (ms)	%DB time
CPU time			97.03		52.28	CPU time			40.26		64.57
db file sequential read	User I/O	1,218	11.76	9.66	6.34	log file parallel write	System I/O	1,854	5.83	3.14	9.35
log file parallel write	System I/O	2,382	7.43	3.12	4.00	SQL*Net message to client	Network	91,222	1.50	0.02	2.40
log file sync	Commit	2,018	7.21	3.57	3.88	latch: shared pool	Concurrency	3	0.62	206.52	0.99
SQL*Net message to client	Network	91,076	1.56	0.02	0.84	os thread startup	Concurrency	3	0.43	142.49	0.69
-os thread startup	Concurrency	6	1.18	197.29	0.64	-log file sync	Commit	35	0.27	7.66	0.43
-						-db file sequential read	User I/O	118	0.11	0.96	0.18

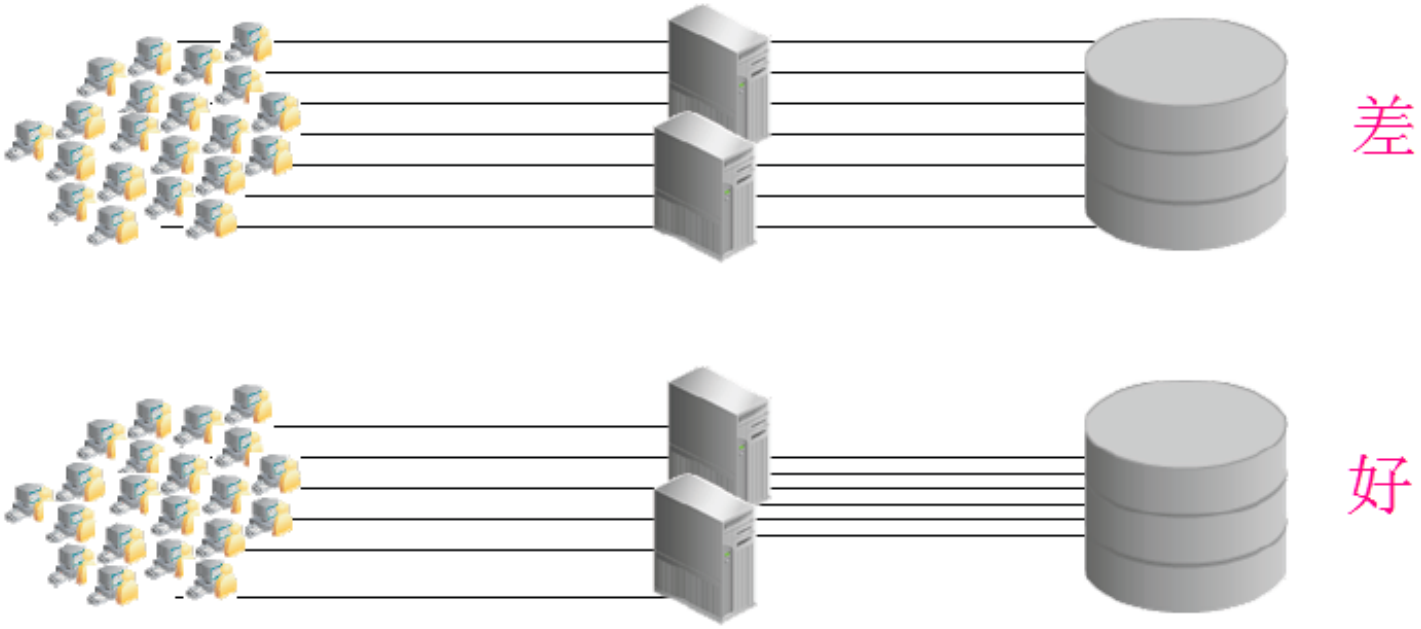
数据库性能基础

- 好的Schema和数据模型设计是好的数据库性能的基础.
- 好的Schema和数据模型设计应该使应用操作更加简单并减少错误发生
  - 选择正确的数据类型(e.g. Char vs VARCAHR2)
  - 使用不同的索引设计高效率的访问路径
  - 选择正确的存储选项
  - 使用分区
  - 收集并维护准确的统计信息
  - 编写简单高效的SQL
  - 尽量避免序列化瓶颈

数据库性能基础

- 应用服务器/中间件管理数据库会话连接，保证数据库的性能和稳定
  - 在一台机器上超过5000进程会使机器变的不稳定
  - 在实验室很容易模拟大量的数据库连接，但是在生产系统上要使用则非常困难
  - logon/logoff对性能的影响进程的建立和撤销开销非常大
  - 连接和去连接到共享内存的开销也非常大
- 共享SQL, 在OLTP应该中使用绑定变量
- 应用应该尽量避免对数据库对象进行creation、drop、truncate等操作

会话管理



- 不要连接所有浏览器用户
- 不要连接、执行操作、断开连接
- 使用连接池—中间层的最大优点

性能与数据安全的一些取舍

- 影响性能的一些数据安全设置:
  - 多镜像控制文件
  - 重做日志组中多镜像日志组成员
  - 频繁Check Point
  - 备份数据文件
  - 归档状态
  - 块校验
  - 并发用户访问和事务

数据库性能基础

CPU and Memory

- For OLTP workloads the CPU utilization should not exceed 65-70%
- Scale Nodes/CPUs/Cores as a power of 2.
- Start with 4Gig Memory per Core and expand according to workload type

I/O

- Target 5-10 millisec response time for disks performing response time critical I/O.
- Start by assuming 30 IOPS per disk for OLTP and 20MBytes per Disk in DSS. This is way below theoretical values but allows for media repair etc.
- You can run HBAs and SAN switches at about 70-90% of theoretical values.
- Allocate and Administer Disk by bandwidth in addition to storage requirements.
- 200MB/s per CPU core

Interconnect

- For Data passing intensive DSS Clusters Balance equalize Interconnect bandwidth with I/O bandwidth. Complex queries running across a cluster will perform sorts and joins across the interconnect.
- In the future we may require higher interconnect bandwidth for DW/BI workloads because the data on disk is compressed
- Do not confuse bits for Bytes when capacity planning Interconnect or SAN.

Oracle的性能解决方案

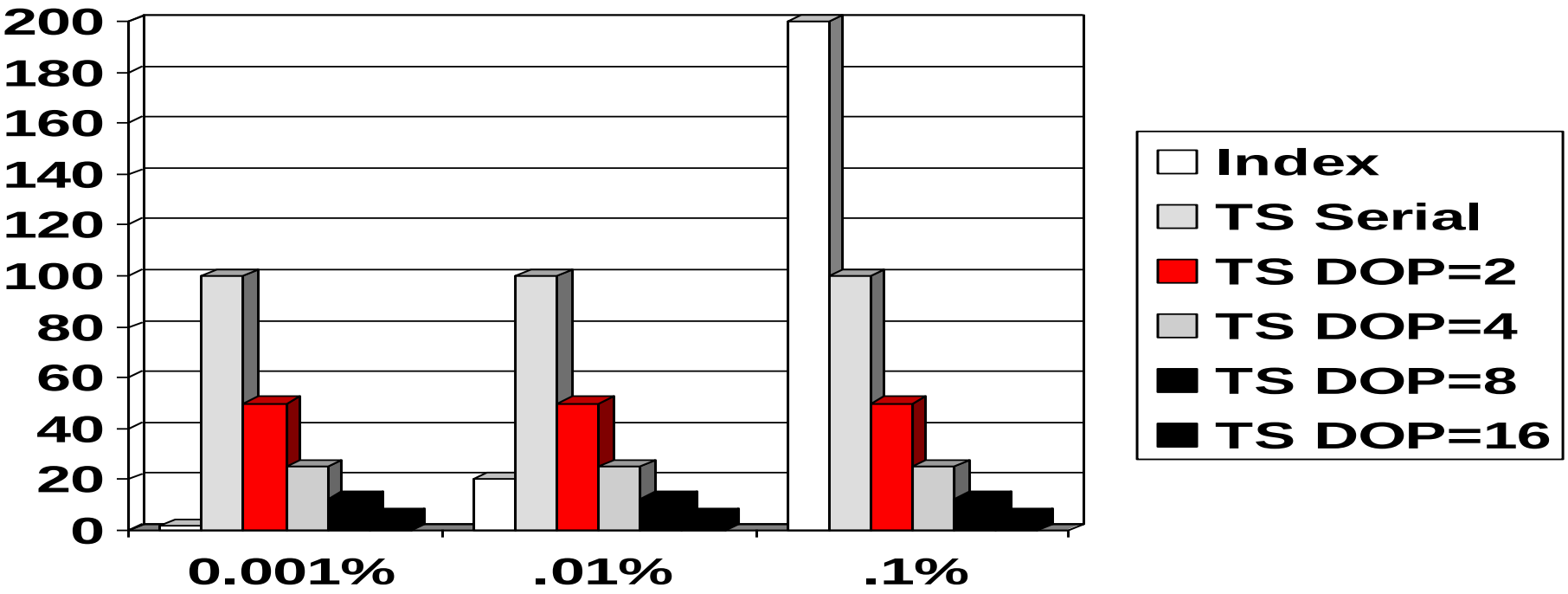
- 绑定变量
- 并行查询
- 索引
- 物化视图
- 分区
- Real Application Cluster
- 压缩
- 结果集缓存
- Active Data Guard
- TimesTen

并行

- 并行查询
  - SELECT /*+ PARALLEL(SALES,9)*/ * FROM SALES;
  - SELECT /*+ PARALLEL_INDEX(c,ic,3)*/ * FROM customers c WHERE cust_city = 'MARSEILLE';
- 并行DML
  - ALTER SESSION ENABLE/DISABLE PARALLEL DML
  - MERGE /*+ PARALLEL(c,3) PARALLEL(d,3) */
    INTO customers c
    USING diff_customers d ON (d.cust_id = c.cust_id)
    WHEN MATCHED THEN
    UPDATE SET c.cust_last_name = d.cust_last_name, c.cust_city = d.cust_city
    WHEN NOT MATCHED THEN
    INSERT (c.cust_id,c.cust_last_name) VALUES (d.cust_id,d.cust_last_name);
- 并行DDL
  - CREATE INDEX
  - CREATE TABLE … AS SELECT
  - ALTER INDEX … REBUILD

并行查询性能分析

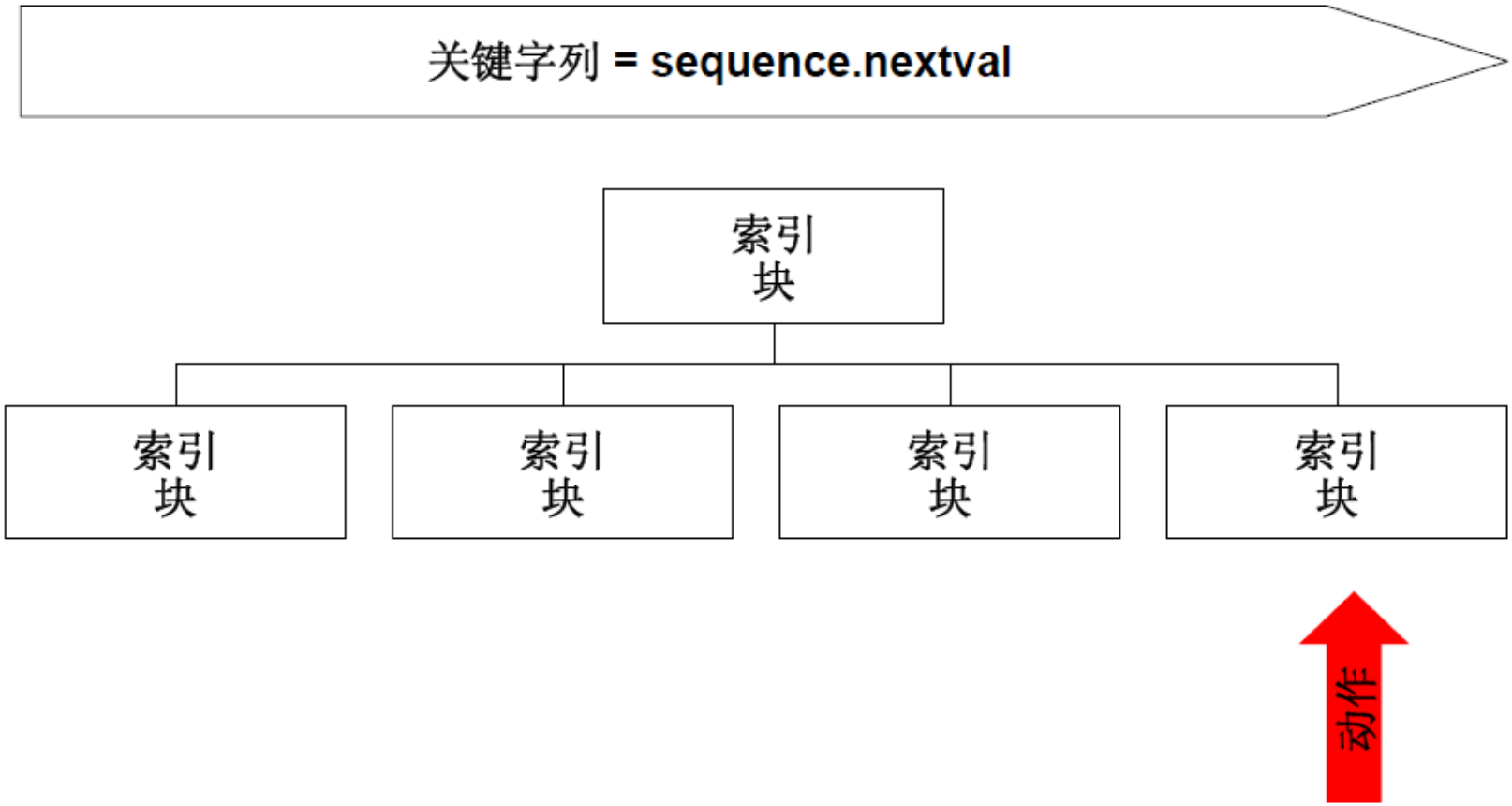
- 假设某张表拥有100,000,000 行记录数，每行记录100 Byte 大小, 表扫描I/O速率是每秒1M/10ms, 索引I/O是8k/5ms并且假设缓存命中率为80%



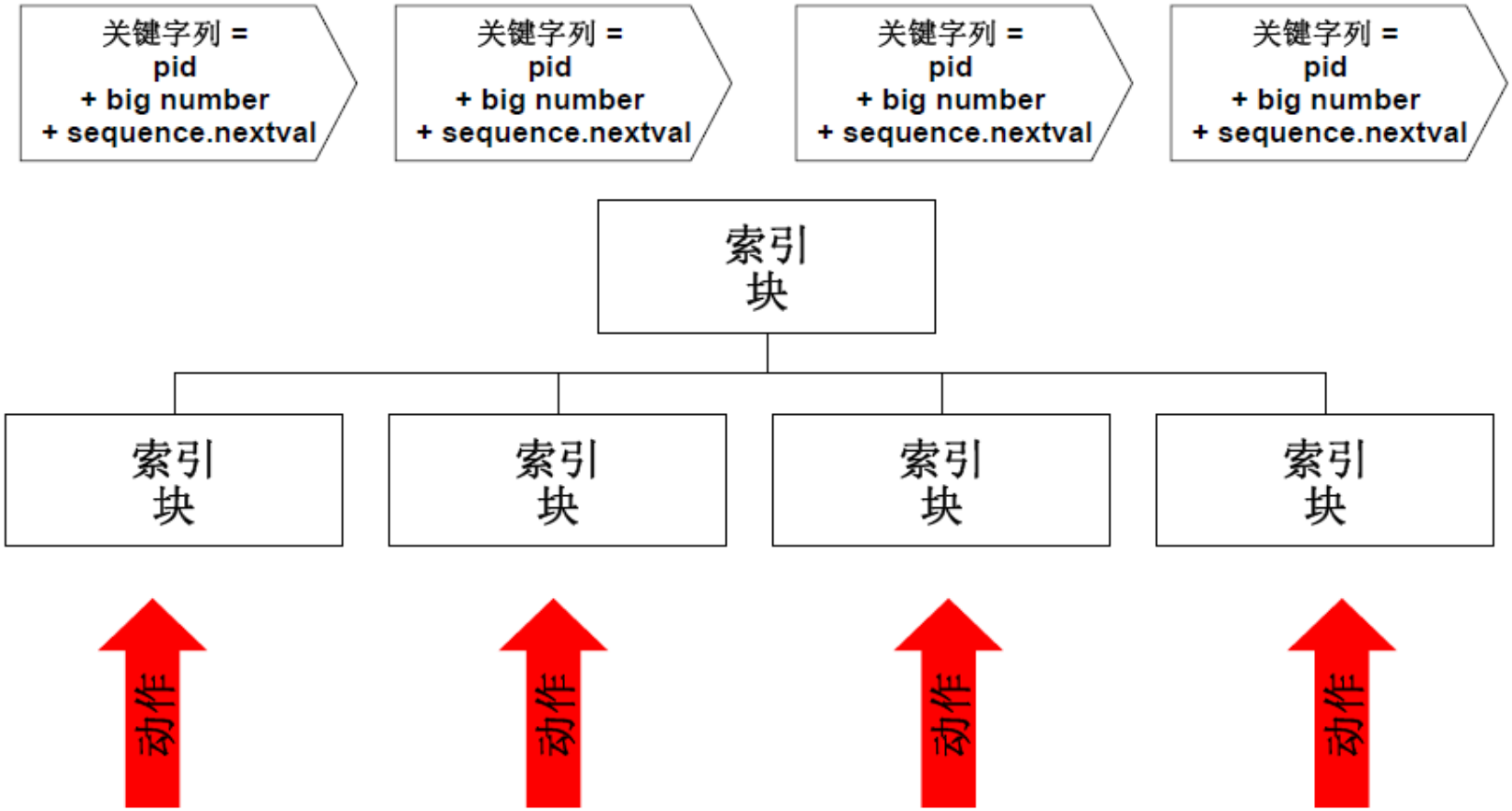
索引

- 索引的影响
  - 加快能适当使用索引的SELECT, UPDATE
  - 减慢INSERT速度
  - 需要额外空间
- 索引使用原则
  - 只有需要的时候才创建相应的索引
  - 为解决某个问题创建一个索引可能会影响其他的SQL执行
  - 删除没用的索引

使用索引：集中IO



使用索引：分散IO



索引的类型

- 唯一及非唯一索引
- 复合索引
- 按索引存储方法
  - B*-tree
    - 普通
    - 反键索引
    - 函数索引
  - 位图索引
- 索引组织表

物化视图

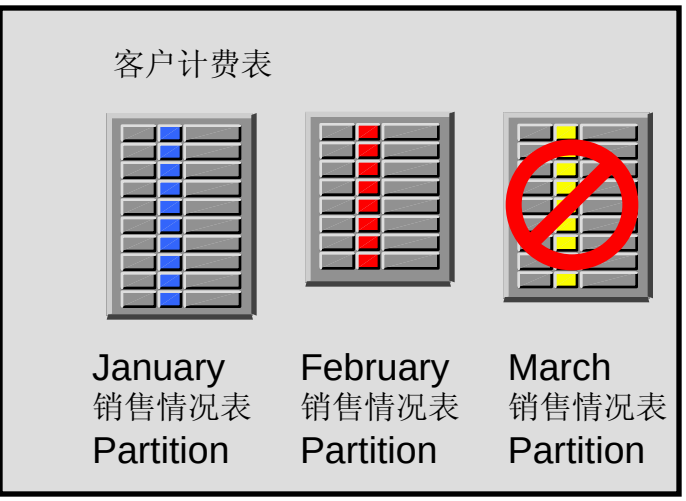
- 物化视图:
  - 是一个预计算的结果集
  - 拥有自己的数据存储:
  - 拥有自己的索引
  - 适用场合:
    - 复杂的join
    - 概要和聚集数据集

分区的作用

- 分区对性能影响
  - 分区忽略
  - 并行处理

数据分区技术大大提高访问速度 — “有的放矢、并行处理”

```
select sum(amount)
From   销售情况表
Where 日期
between 20-JAN-2004
and     5-FEB-2004
```

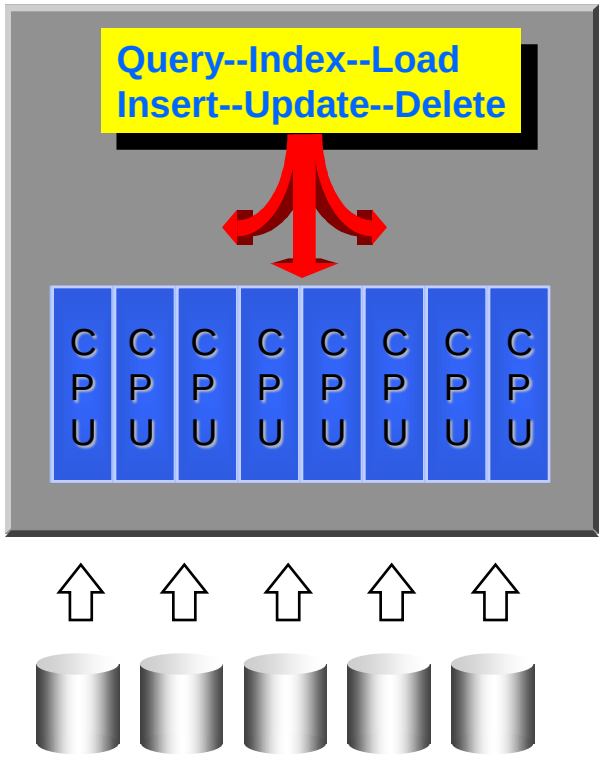


- 分区可以显著提高访问大表时的 性能
- 分区的存在对应用系统是透明的

分区忽略技术

系统核心自动根据分区情况优化数据访问，忽略无关的数据分区

强大的并行处理能力



- 随着数据库的增长，必须采用并行处理方式以保证响应时间
- 在数据库中进行并行处理可以显著提高批量操作的性能
  - 批量更新
  - 批量删除
  - 批量插入
- 并行执行 SQL语句
- 数据动态分片
- RAC的多节点并行更新

EXPLAIN PLAN命令: 示例1

```
EXPLAIN PLAN
SET STATEMENT_ID = 'demo01' FOR
SELECT e.last_name, d.department_name
FROM hr.employees e, hr.departments d
WHERE e.department_id = d.department_id;

Explained.
```

EXPLAIN PLAN 命令并不实际执行该SQL。

EXPLAIN PLAN命令: 示例2

```
SELECT PLAN_TABLE_OUTPUT FROM TABLE(DBMS_XPLAN.DISPLAY());
```

```
Plan hash value: 2933537672

-------------------------------------------------------------------------------
| Id  | Operation                     | Name        | Rows  | Bytes | Cost (%CPU|
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |             |   106 |  2862 |     6  (17|
|   1 |  MERGE JOIN                   |             |   106 |  2862 |     6  (17|
|   2 |   TABLE ACCESS BY INDEX ROWID| DEPARTMENTS |    27 |   432 |     2   (0|
|   3 |    INDEX FULL SCAN            | DEPT_ID_PK  |    27 |       |     1   (0|
|*  4 |   SORT JOIN                   |             |   107 |  1177 |     4  (25|
|   5 |    TABLE ACCESS FULL          | EMPLOYEES   |   107 |  1177 |     3   (0|
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")
       filter("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")

18 rows selected.
```

SQL*Plus中设置AUTOTRACE: 示例1

- 启动 AUTOTRACE

```
set autotrace on
```

- 隐藏输出

```
set autotrace traceonly
```

- 只显示执行计划
  - AUTOTRACE需要实际执行SQL语句

```
set autotrace traceonly explain
```

SQL*Plus中设置AUTOTRACE: 示例2

```
set autotrace traceonly statistics

SELECT *
FROM products;
```

```
Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
          9  consistent gets
          3  physical reads
          0  redo size
      15028  bytes sent via SQL*Net to client
        556  bytes received via SQL*Net from client
          6  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
         72  rows processed
```

哪些因素影响执行计划

- 统计信息
  - 对象统计信息
    - 表
    - 列
    - 索引
  - 系统统计信息
    - I/O 性能
    - CPU 性能
- 优化器、Hints
- 初始化参数
- 索引
- 分区
- 物化视图
- 并行
- … …

统计信息如何收集

- 自动收集统计信息
  - GATHER_STATS_JOB
- 手动收集统计信息
  - 使用DBMS_STATS 包
- 动态采样
  - optimizer_dynamic_sampling

优化器

- RBO
- CBO
  - ALL_ROWS
  - FIRST_ROWS
  - FIRST_ROWS_n

Hints使用举例

```
UPDATE /*+ INDEX(p PRODUCTS_PROD_CAT_IX)*/
products p
SET   p.prod_min_price =
        (SELECT
         (pr.prod_list_price*.95)
           FROM products pr
           WHERE p.prod_id = pr.prod_id)
WHERE p.prod_category = 'Men'
AND   p.prod_status = 'available, on stock'
/
```

Hints类型

- 优化器模式
  - ALL_ROWS
- 访问路径
  - FULL
- 查询转换
  - STAR_TRANSFORMATION
- Join顺序
  - ORDERED
- Join操作
  - USE_NL， USE_HASH
- 其他
  - APPEND

SQL Trace

- 针对当前会话:

```
SQL> ALTER SESSION SET sql_trace = true;
```

- 针对任意会话:

```
SQL> EXECUTE dbms_session.set_sql_trace(true);
```

```
SQL> EXECUTE dbms_system.set_sql_trace_in_session
  2  (session_id, serial_id, true);
```

- 实例级，可以直接设置初始化参数:

```
SQL_TRACE = TRUE
```

TKPROF格式化trace文件

- TKPROF 命令示例:

```
OS> tkprof tracefile outputfile [options]
```

```
OS> tkprof
OS> tkprof ora_902.trc run1.txt
OS> tkprof ora_902.trc run2.txt sys=no sort=execpu print=3
```

TKPROF 格式化后输出：示例

```
...
select max(cust_credit_limit)
from customers
where cust_city ='Paris'


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  -------
Parse        1      0.02       0.02          0          0          0        0
Execute      1      0.00       0.00          0          0          0        0
Fetch        2      0.10       0.09       1408       1459          0        1
------- ------  -------- ---------- ---------- ---------- ----------  -------
total        4      0.12       0.11       1408       1459          0        1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 61

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=1459 pr=1408 pw=0 time=93463 us)
     77   TABLE ACCESS FULL CUSTOMERS (cr=1459 pr=1408 pw=0 time=31483 us)
```

10046事件分析

```
SQL> alter session set events '10046 trace name context forever,
level 8';
Session altered.
SQL> alter session set events '10046 trace name context off';
Session altered.
```

```
执行跟踪:
SQL> exec dbms_system.set_ev(5,323,10046,8,'scott');
PL/SQL procedure successfully completed.
结束跟踪:
SQL> exec dbms_system.set_ev(5,323,10046,0,'scott');
PL/SQL procedure successfully completed.
```

10053事件分析

```
SQL> alter session set events '10053 trace name context forever,
level 1';
Session altered.
SQL> alter session set events '10053 trace name context off';
Session altered.
```

```
执行跟踪:
SQL> exec dbms_system.set_ev(5,323,10053,1,'scott');
PL/SQL procedure successfully completed.
结束跟踪:
SQL> exec dbms_system.set_ev(5,323,10053,0,'scott');
PL/SQL procedure successfully completed.
```

QUESTIONS & ANSWERS