
ORACLE
OPEN
WORLD

ORACLE

Real World Performance Part I

Andrew Holdsworth
Director of Real World Performance Server Technologies

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.
The development, release, and timing of any features or functionality described for Oracle's products remain at the sole discretion of Oracle.

Real World Performance Part I

- How to use today's sessions
- Feedback from last year
- Some performance basics
- Gathering data from your systems
- Key Generation
- Shared Servers Revisited

How to Use Today's Sessions

- Part I
  - Focus on the “lower half” below the SQL Interface
  - Debugging and monitoring skills
  - Other system performance updates
- Part II
  - Focus on all SQL related issues
  - Optimizer
  - Schema statistics
  - Debugging SQL Execution Plans
- Part III
  - Advanced Features and Exotics
  - More Time for Questions

Panel Questions

- Please bring questions to stage or Francoise before and during the session
- Please include the following
  - Name, Company
  - Computing environment if relevant
  - Database Version
  - Question !
- Please note we will not be performing debugging operations in the panel sessions. Please focus on the subject matter of Real World Performance and not current issues/war stories.

Some Feedback From Last Year

- User confused what to do when a query does not return immediately ?
- What is a good plan ?
- What is a serialization problem ?

Some Performance Basics

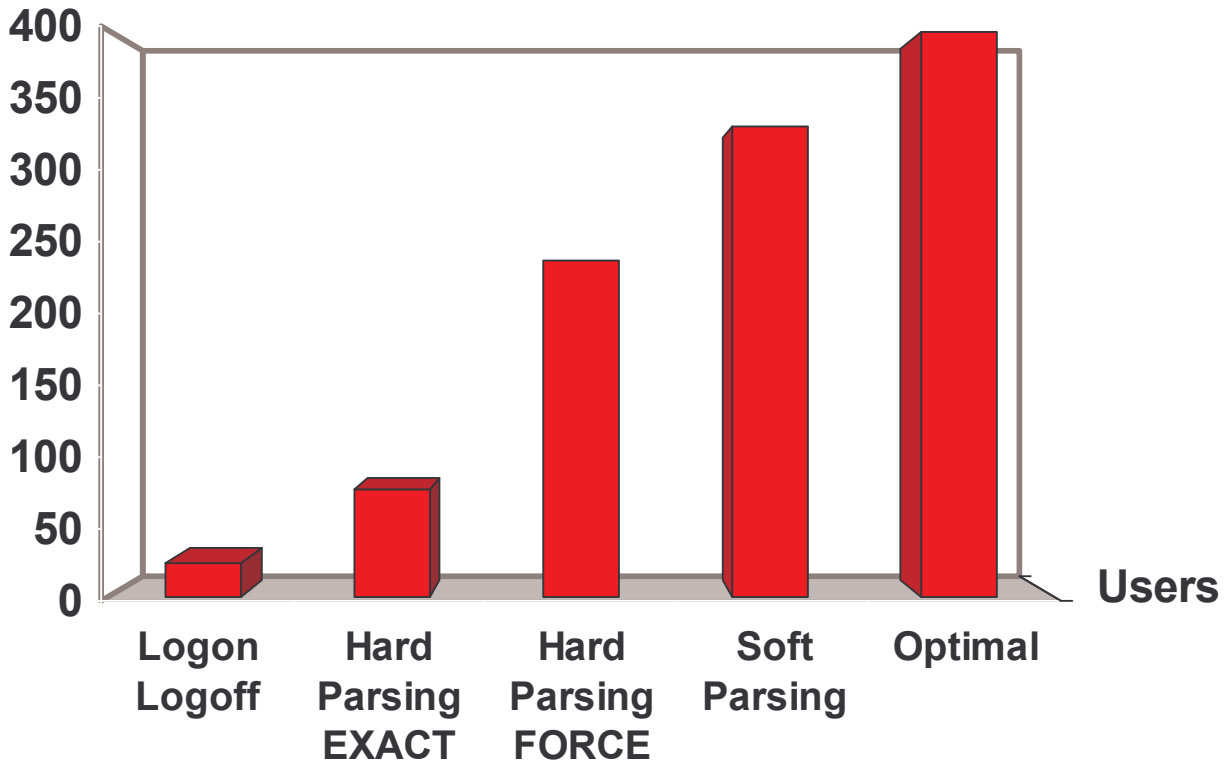
- Definitions
  - Serialization What is It ?
    - One or more processes waiting for shared resources on a system.
  - Process State
    - Hung – Process halted cause unknown
    - Blocked – Process halted waiting for a resource to become free e.g. Transaction lock.
    - Waiting – Process waiting for an event outside the database kernels control e.g. I/O, semaphore
    - Running – Process executing database work
    - Spinning – Process looping out of control burning CPU

Some Performance Basics

- Machine/Process Utilization
  - CPU Bound
    - Time spend in usr/sys/wio/idle
  - Disk Bound
    - Time queued/seeking/transfering
  - Network Bound
    - Time spend waiting for bytes to be sent/received over a network
  - Application Server Bound
    - Middle tier issues preventing more database calls

Some Performance Basics

- Sessions, Cursors, Arrays, SQL, Etc.



Monitoring your System

Graham Wood - Architect

Gathering Database Instance and Operating System Data from your Systems

- What data should we be gathering and keeping
  - Prior to 10g, or 10g without Diagnostic Pack
    - Statspack at hourly intervals (can use spauto.sql)
    - System level performance data (e.g. spooled vmstat, ps)
    - Keep data for time period longer that your business cycle, month, quarter etc. It may require manual purging
  - 10g with Diagnostic Pack
    - AWR data captured and purged automatically
    - One hour capture interval good in most cases
    - Increase retention period to longer than business cycle, month, quarter etc. Default is only 7 days.
    - Although more OS level data in AWR still capture occasional ps output particularly if running Shared Servers or other software on the same machine

Time based approach to Instance Analysis

- With the diagnostic pack ADDM will do the analysis for you
- Otherwise study AWR or Statspack reports
- Manual method:

  Determine is where the time is spent on the database server(s)
  - Waiting for system and shared resources
  - Time executing by SQL statement, Application Module, Service Etc.
- Use this data to determine ultimate bottlenecks

Key Generation Issues

Andrew Holdsworth

Key Generation

- What has made this so important
  - Faster CPUS
  - Increased concurrency
  - More Insert orientated applications
    - Travel bookings
    - Market trades
    - Telco calls, messages, events
  - Memory access speeds and concurrency have not kept pace with CPU processing speeds

Key Generation

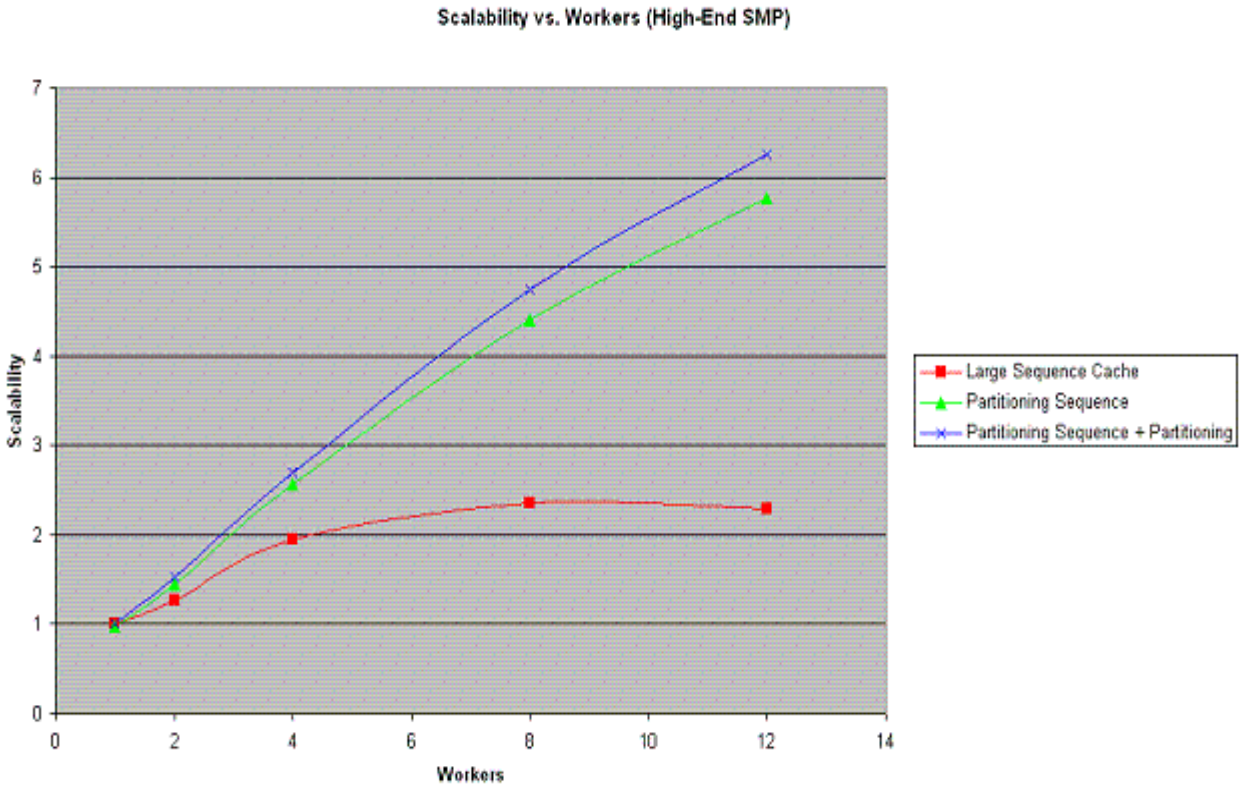
- Contention point description
  - Unique/Primary Key Generation from a sequence yields an increasing key value.
  - This results in a contention point in the right hand side of the B-Tree Unique index used to enforce the uniqueness required for integrity.
  - This problem has been seen more and more as faster CPUs process more rows and compete for the same memory structures.
  - A leaf block goes where a leaf block goes.

Contention Management Methods available

- Hash Partition on the unique key to create a series warm indexes rather than one hot index.
  - Works well for insert workloads
  - This solution does not work well if range scanning on the primary key and multiple local index probes are required.
- Reverse Key Index
  - Works well for Indexes that fit inside the buffer cache.
  - On large indexes greater than the size of the buffer cache this will flood the buffer cache with the index and make insert operations disk bound.
- Design a specific key generation mechanism
  - Ensures uniqueness
  - Localizes Index maintenance to single insert process avoiding memory/block contention
  - Allows index maintenance to remain in memory and avoid I/O

Key Generation

- Lab work tests on custom key generation to avoid contention



More Details on Custom Sequence Generation

- Evaluating a prototype that allows the sequence to be prefixed by either or session id and Instance number automatically.
- This would allow existing applications to run and scale better with no code changes !
- See Bug #5584365 for more details.

Shared Servers Revisited

Sumanta Chatterjee

Gathering Shared Server Performance

- Monitor and collect data every 10-15 seconds
- v$queue – monitor ss and dispatcher queues
  - Aggregate by queue type
  - queued – number of requests currently queued
  - wait – total wait time in hundredths of a second
  - totalq – total requests queued; compute queue rate
- v$shared_server / v$dispatcher
  - status
    - SS – EXEC, COMMON(IDLE), WAIT (ENQ), etc.
    - Dispatcher: WAIT, SEND, CONNECT, etc.
  - messages – total messages processed
  - busy:idle ratio gives indication of %busy

Gathering Shared Server Performance

- v$system_event (Top 5 Timed Events)
  - What are the resources showing contention?
  - Any lock (TX) contention?

Shared Server: Case Study I

- Shared Server required to
  - Reduce number of processes
  - 10-20K sessions meant 10-20K processes if using dedicated
- Problem symptom
  - CPU usage was very high
  - No contention in v$views
- Solution
  - Code path reductions (bug5362897 – 9.2.0.5 one-off patch)
  - Code optimizations (bug5347812 – 9.2.0.5 one-off patch)
- Result
  - CPU reduction from 90-95% to less than 50% on a 24 cpu system

Shared Server: Case Study II

- Shared Server required to
  - reduce number of processes
  - reduce overhead of repeated session logon/logoff
- Problem symptom
  - 2x increase in application workload led to slow performance
  - v$queue showed VCs queued
  - v$shared_server showed idle SS
  - “virtual circuits *” latch contention
  - Application TX enqueue contention
- Solution (bug5150366 - 9.2.0.7 one-off patch)
  - Granularized queues
  - Eliminate hot spots
- Result
  - Accommodated continued increase in application workload

Panel

Juan Loaiza

Graham Wood

Cecilia Gervasio

Sumanta Chatterjee

Andrew Holdsworth

For More Information

http://search.oracle.com

or

http://www.oracle.com/


ORACLE

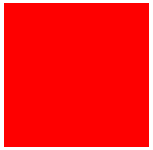

ORACLE
OPEN
WORLD

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.
The development, release, and timing of any features or functionality described for Oracle's products remain at the sole discretion of Oracle.

ORACLE

Real-World Performance Roundtable, Part II
The Optimizer, Schema Statistics and SQL Tuning

Andrew Holdsworth, Greg Rahn,
Mohamed Zait, Mohamed Ziauddin, Hermann Baer

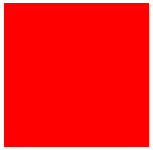


Real World Performance Part II

- Optimizer
- Schema statistics
- SQL execution plans

Panel Questions

- Please bring questions to stage or Francoise before and during the Session
- Please include the following
  - Name, Company
  - Computing environment if relevant
  - Database Version
  - Question !
- Please note we will not be performing debugging operations in the panel sessions. Please focus on the subject matter of Real World Performance and not current issues/war stories.



Agenda

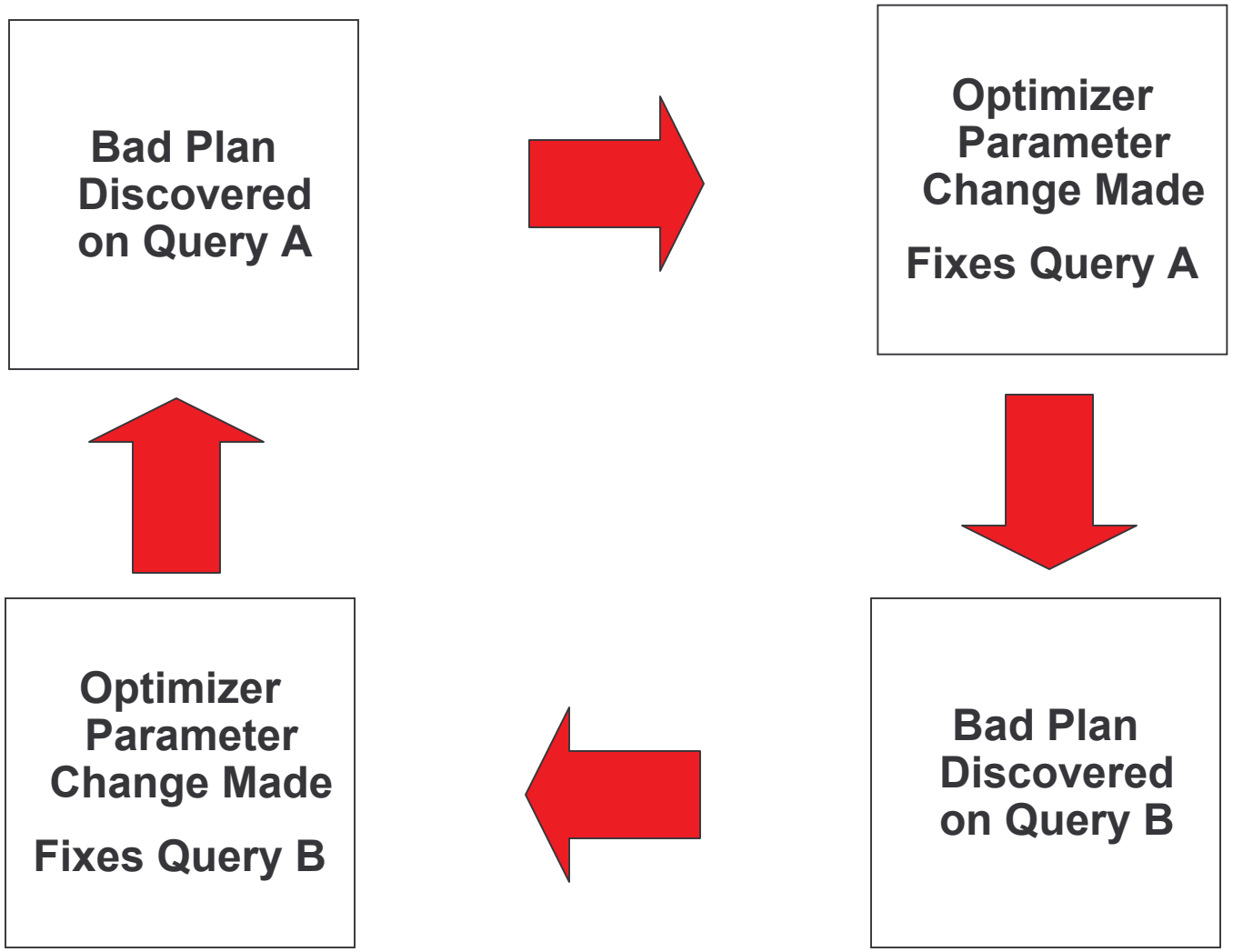
- Optimizer Myths
- 90-9-1 Stats Methodology
  - 90% of the time the default sample works
  - 9% of the time a larger sample works
  - 1% of the time the sample size is irrelevant
- Checking Cardinality

Optimizer Myths

Optimizer Myths

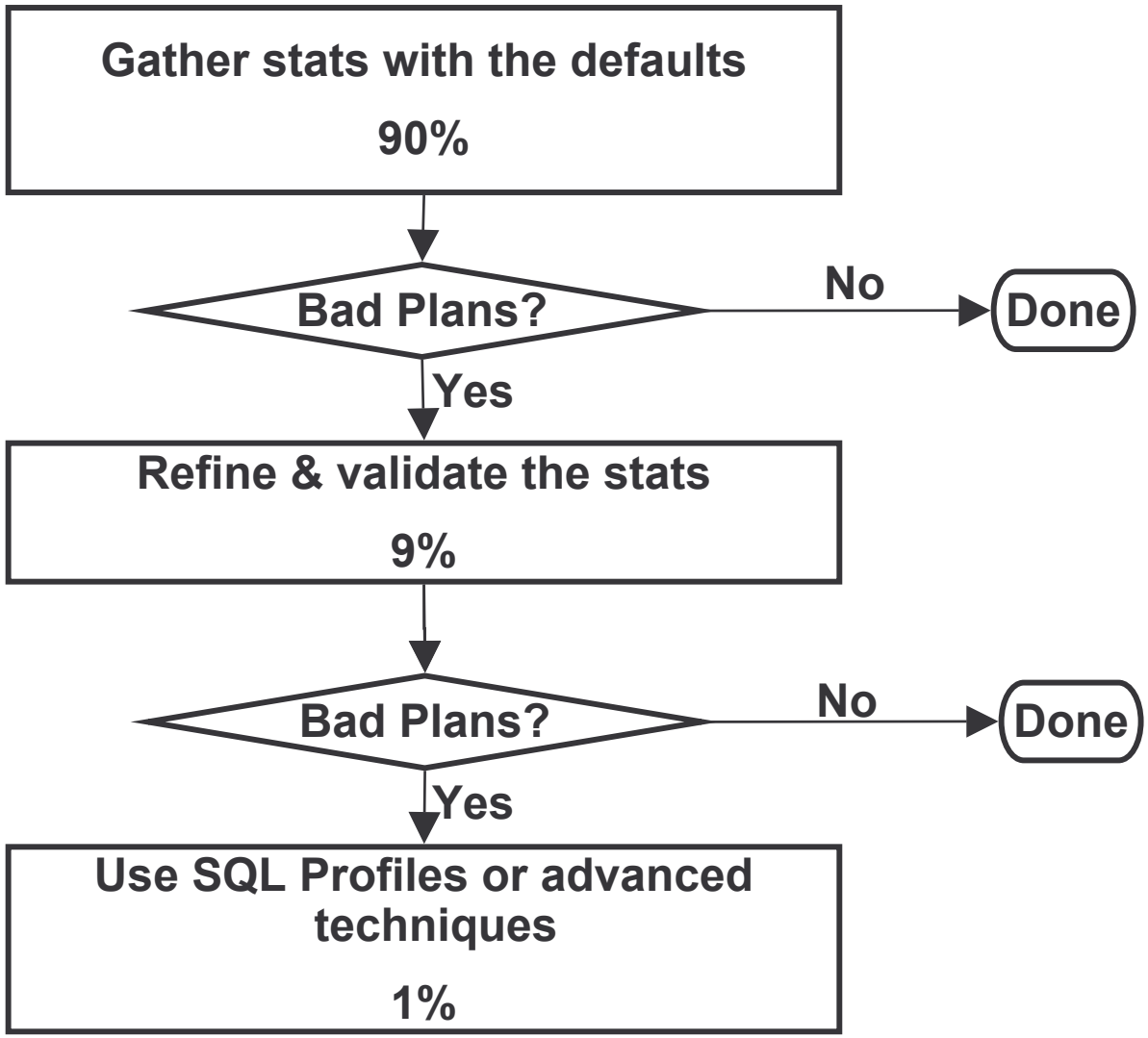
- Need to tune or design for the optimizer
- Order of the predicate can change the plan
- A compute statistics is required to get accurate statistics
- A minimum 10% sample is required to get accurate statistics
- Numerous hidden init.ora parameters needed to yield good plans
- OPTIMIZER_* init.ora parameter need to be tuned
- 10053 trace required to debug bad plans during initial triage

The “Parameter Change” Fan Trap



90 – 9 –1 Stats Methodology

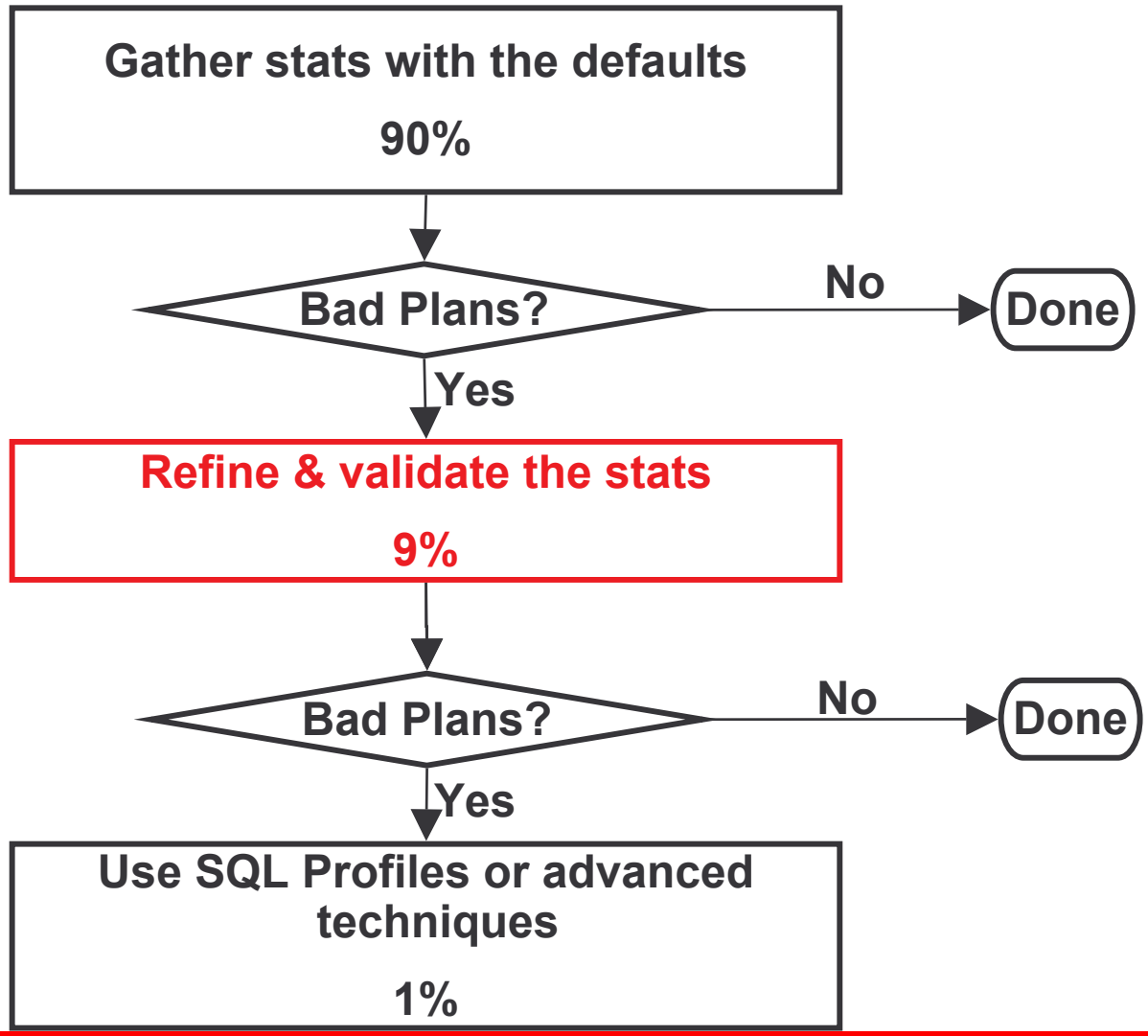
90-9-1 Stats Methodology Flowchart



How the Optimizer Generates Good Plans: The 90%

- Stats collected on schema
- SQL workload run against schema
  - DML information collected (dba_tab_modifications)
  - Predicate information collected (sys.col_usage$)
- Stats regathered on schema
  - Automatically – 10% change & automatic stats gathering job
  - Manually – when the DBA deems fit
- Stats and plans evolve with data change and usage
  - Predicate information collected at run time can result in histograms being created on next execution of dbms_stats.gather_*

90-9-1 Stats Methodology Flowchart



Refining The Stats

- When a new or larger sample may be needed
  - Data skew
  - Out-of-range
- Ways stats can be refined
  - Regather manually on less than 10% change
  - Increase the sample size for a given table or index
  - Compute statistics
  - Create histograms
- Validate the stats
  - Check the cardinality in the explain plan
  - Verify values in the data dictionary views

Refining The Stats Example (1 of 2)

```
SQL> exec dbms_stats.gather_table_stats(user,'SKEW1');
SQL> select column_name, num_distinct, sample_size
     from user_tab_col_statistics where table_name='SKEW1'
```

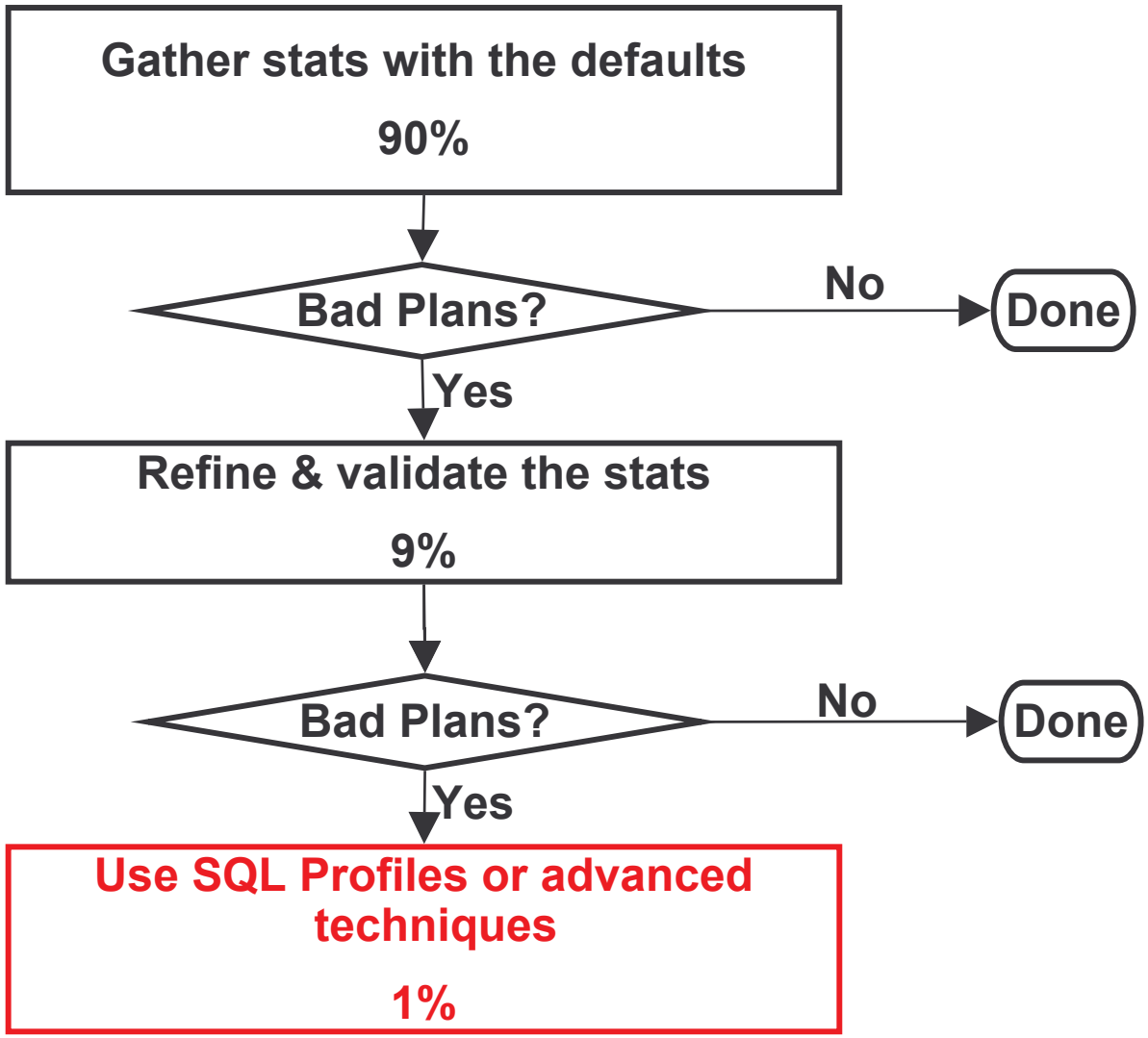
COLUMN_NAME	NUM_DISTINCT	SAMPLE_SIZE
C1	2,092	7,703
C2	410,792	777,038
C3	61,527	77,883
C4	2	7,703
C5	2	7,703

Refining The Stats Example

Column	Auto Sample	1% Sample	10% Sample	30% Sample	50% Sample	Actual
C1	2,092	11,305	31,402	46,890	56,282	60,351
C2	410,792	260,447	607,423	897,853	1,102,326	1,289,760
C3	61,527	103,083	358,233	567,937	685,271	777,942
C4	2	2	2	2	2	2
C5	2	2	2	2	2	2

Statistics need to be representative, not exact

90-9-1 Stats Methodology Flowchart



When Statistics Alone Aren't Enough

- Issues
  - Correlated columns / multi-column predicates
  - Functions
  - Temporary tables
- Options
  - SQL Profiles
  - Hints – cardinalty, dynamic_sampling

Correlated Columns Example: Zodiac Signs (1 of 4)

- Test Case: Table of birthdays & Zodiac signs
  - Approx. 12 million rows [actual: 11,960,320]
  - 32,768 birthdays per day of the month
- How many people have December birthdays?
  - ~1,000,000 (1/12th)

```
select count(*) from zodiac where birth_date between
to_date('2006-DEC-01','yyyy-mon-dd') and to_date('2007-JAN-01','yyyy-mon-dd')

-----------------------------------------------------------
| Id  | Operation           | Name    | E-Rows | A-Rows |
-----------------------------------------------------------
|   1 |  SORT AGGREGATE     |         |      1 |      1 |
|*  2 |   TABLE ACCESS FULL| ZODIAC  |   1018K|   1015K|
-----------------------------------------------------------
```

Correlated Columns Example: Zodiac Signs (2 of 4)

- How many people are of the Zodiac sign Sagittarius?
  - ~1,000,000 (1/12th)

```
select count(*) from zodiac
where zodiac_sign = 'sagittarius'

-----------------------------------------------------
| Id  | Operation           | Name    | E-Rows | A-Rows |
-----------------------------------------------------
|   1 |  SORT AGGREGATE     |         |      1 |      1 |
|*  2 |   TABLE ACCESS FULL| ZODIAC  |   1011K|    983K|
-----------------------------------------------------
```

Correlated Columns Example:
Zodiac Signs (3 of 4)

- Question: How many people are born in December and have the Zodiac sign Sagittarius?
  - Answer: Without knowing anything about the Zodiac we would guess 1/12 * 1/12 = 1/144 or ~83,333.
  - Correct Answer: 720,896

    The reality is that 22 of the 31 days of December are of the sign Sagittarius. The statistical estimate is inaccurate due to data correlation.

Correlated Columns Example:
Zodiac Signs (4 of 4)

```
select count(*) from zodiac where birth_date between
to_date('2006-DEC-01','yyyy-mon-dd') and
to_date('2007-JAN-01','yyyy-mon-dd')
and zodiac_sign = 'sagittarius'

-----------------------------------------------------
| Id  | Operation          | Name    | E-Rows | A-Rows |
-----------------------------------------------------
|   1 |  SORT AGGREGATE    |         |      1 |      1 |
|*  2 |   TABLE ACCESS FULL| ZODIAC  |  83713 |   720K |
-----------------------------------------------------
```

Checking Cardinality

Sample Query

```
select …
from   a, b, c, d
where  a.exloan_id = b.exloan_id and
       a.state = c.state_code and
       c.single_fam_st_code = d.index_code and
       b.reporting_month =
         to_date ('1992-11-01', 'yyyy-mm-dd') and
       d.period =
         to_char (b.reporting_month, 'yyyyq')
```

DBMS_XPLAN.DISPLAY

```
SQL> explain plan for <query>;
SQL> select * from table(dbms_xplan.display);
```

```
-------------------------------------------------------------------------------
| Id  | Operation                    | Name        | Rows  | Pstart| Pstop |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |             |  4122 |       |       |
|*  1 |  HASH JOIN                   |             |  4122 |       |       |
|*  2 |   HASH JOIN                  |             |  4852 |       |       |
|   3 |    TABLE ACCESS FULL         | C           |    51 |       |       |
|   4 |    PARTITION HASH ALL        |             |  4874 |     1 |    16 |
|*  5 |     HASH JOIN                |             |  4874 |       |       |
|   6 |      PARTITION RANGE SINGLE  |             |  4874 |    35 |    35 |
|*  7 |       TABLE ACCESS FULL      | B           |  4874 |   545 |   560 |
|   8 |      TABLE ACCESS FULL       | A           |  7469K|     1 |    16 |
|   9 |   TABLE ACCESS FULL          | D           |   512K|       |       |
-------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   1 - access("D"."INDEX_CODE"="C"."SINGLE_FAM_ST_CODE" AND
              "D"."PERIOD"=TO_NUMBER(TO_CHAR(INTERNAL_FUNCTION("B"."REPORTING_MONTH"),'yyyyq')))
   2 - access("A"."STATE"="C"."STATE_CODE")
   5 - access("A"."EXLOAN_ID"="B"."EXLOAN_ID")
   7 - filter("B"."REPORTING_MONTH"=TO_DATE('1992-11-01 00:00:00', 'yyyy-mm-dd hh24:mi:ss'))
```

DBMS_XPLAN.DISPLAY_CURSOR

```
SQL> alter session set statistics_level=all;
SQL> <execute query>
SQL> select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

Id	Operation	Name	Starts	E-Rows	A-Rows
*1	HASH JOIN		1	4122	4874
*2	HASH JOIN		1	4852	4874
3	TABLE ACCESS FULL	C	1	51	51
4	PARTITION HASH ALL		1	4874	4874
*5	HASH JOIN		16	4874	4874
6	PARTITION RANGE SINGLE		16	4874	4874
*7	TABLE ACCESS FULL	B	16	4874	4874
8	TABLE ACCESS FULL	A	16	7469K	7477K
9	TABLE ACCESS FULL	D	1	512K	512K

```
Predicate Information (identified by operation id):
---------------------------------------------------
1 - access("D"."INDEX_CODE"="C"."SINGLE_FAM_ST_CODE" AND
 "D"."PERIOD"=TO_NUMBER(TO_CHAR(INTERNAL_FUNCTION("B"."REPORTING_MONTH"),'yyyyq')))
2 - access("A"."STATE"="C"."STATE_CODE")
5 - access("A"."EXLOAN_ID"="B"."EXLOAN_ID")
7 - filter("B"."REPORTING_MONTH"=TO_DATE('1992-11-01 00:00:00', 'yyyy-mm-dd hh24:mi:ss'))
```

90-9-1 Methodology Recap

- Representative stats generally yield good plans
- There are cases where stats need more granular gathering
- There are cases where stats alone are not enough
- Use cardinality estimates as initial triage

Panel

Hermann Baer

Mohamed Zait

Mohamed Ziauddin

Greg Rahn

Andrew Holdsworth

Questions & Answers

For More Information

http://search.oracle.com

or

http://www.oracle.com/


ORACLE


ORACLE
OPEN
WORLD

ORACLE

Real World Performance Part III

Andrew Holdsworth
Director of Real World Performance Server Technologies

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.
The development, release, and timing of any features or functionality described for Oracle's products remain at the sole discretion of Oracle.

Real World Performance Part III

- New areas of DB functionality not covered in previous years
  - PL/SQL
  - XML
  - Text
  - Streams
- General performance Questions not covered in Parts I, II or III

Panel Questions

- Please bring questions to stage or Francoise before and during the Session
- Please include the following
  - Name, Company
  - Computing environment if relevant
  - Database Version
  - Question !
- Please note we will not be performing debugging operations in the panel sessions. Please focus on the subject matter of Real World Performance and not current issues/war stories.

PL/SQL

Bryn LLewellyn

PL/SQL Performance Fundamentals

- PL/SQL 10.x is 2 x Fast as PL/SQL 9.2
  - Upgrade is the easiest speedup for PL/SQL intensive programs
- Ask: “Is PL/SQL really the problem?”
  - PL/SQL is often the carrier for SQL statements that dominate the system resources and elapsed time of any executing program. Improving the SQL usually yields the biggest payback.
- Make the extra coding effort to use bulk PL/SQL operations
  - Inserts
  - Fetches
  - Bulk operations performance gains can exceed the gains achieved by upgrade to 10g+.
- Evaluate the use of native compilation
- Dynamic SQL is fine *per se* – but it does give you increased power to code poorly performing approaches!

Streams

Patricia McElroy

Software Updates for Streams

- 10.2.0.2 Recommended Patches:
  - 5563854 Merge patch for Capture
  - 5339787 Required Checkpoint SCN patch
  - 5113125 Merge patch for Apply

Watch OTN Streams website for custom software

- http://otn.oracle.com/products/dataint/content.html

General Configuration Tips

- Separate queue for each capture and apply, also each source database
- Init.ora:

  _job_queue_interval=1

  _spin_count= 5000 (or greater)

  Streams_pool_size=200M

- For WANs: (SQLNET parameters)
  - Increase SDU (sqlnet.ora, tnsnames.ora, listener.ora)
  - Increase send_buf_size, recv_buf_size

10gR2 Streams Process Parameters

Capture: (DBMS_CAPTURE package)

- Set retention time for capture checkpoints as needed
  Alter_capture('captureName', checkpoint_retention_time=>7)
- Reduce the capture checkpoint frequency parameter
  Set_parameter('captureName','_checkpoint_frequency','100')

Propagation

- Use queue_to_queue parameter set to TRUE
  - Source and target must be 10.2 or above

Apply: (DBMS_APPLY package)

- Set_parameter('applyName','parallelism','4')
- Set_parameter('applyName','_dynamic_stmts','Y')
- Set_parameter('applyName','_hash_table_size','10000000')
- Set_parameter('applyName','_txn_buffer_size','10+*parallelism*')
- Set_parameter('applyName','disable_on_error','N')

Apply Performance Tips

- ALTER TABLE SYS.STREAMS$_APPLY_PROGRESS INITRANS 16 PCTFREE 10;

- Batch Processing
  - Frequent commits (txn size < 1000)
  - Consider procedural replication (sample code available)

Apply CLOB Performance

For tables with CLOB columns (10.2 only),

- Register error handler for table UPDATE, DELETE, LOB_UPDATE

```
exec dbms_apply_adm.set_dml_handler
 ('HR.TABLE_ONE','TABLE','UPDATE',true,'STRADM.DML_FUN',
     assemble_lobs=> true);
```

- At minimum, Error Handler should do LCR.EXECUTE

```
create or replace procedure dml_fun(lcr_anydata in sys.anydata)
   authid current_user is
      lcr   sys.lcr$_row_record;
      dummy pls_integer;
   begin
      dummy := lcr_anydata.getObject(lcr);
      lcr.execute(true);
   end;
```

More on Streams….

- Visit Oracle Streams booth in Exhibit Hall, Moscone West D20
- Other Streams presentation:
  - Implementing Replication with Streams (S281220)
    - Moscone South - Room 306 South on **Thursday** 8:00am
- Look for Sample Code for Streams on OTN
- Bookmark OTN page Oracle Information Integration features:
  http://otn.oracle.com/products/dataint/

XML

Nipun Agarwal

Schema Registration & Loading

- For faster update and queries, use xdb:maintainDOM=false if DOM fidelity not required
- Set enable_hierarchy_param to ENABLE_HIERARCHY_NONE if repository usage not required
- Loading large schema based documents through ftp/dav is faster
- Set XDBCORE-LOADABLEUNIT-SIZE=>default 16Kb
- Set XDBCORE-XOBMEM-BOUND=>default 1024Kb

Storage & Query

- If a child element/attribute of a collection needs to be accessed or updated, store the collection as a nested table (xdb:storeVarrayAsTable="true")
- If using nested tables consider using heap tables (default is IOT)
- For efficient query performance, avoid following schema constructs where possible:
  - Recursive type definitions
  - <any> and anytype
  - Usage of xdb:sqlType = "CLOB" for elements whose type is complex

Query

- For better query performance, avoid the following Xpath constructs where possible
  - Minimize using descendent axis, reverse axis, following axis, sibling axis
  - Wildcards (except under certain restrictive conditions)
  - Xpath functions other than Oracle extension functions and the following for: not, floor, ceiling, substring, string-length, translate
  - UNION operations

Repository

- Share ACLs amongst resources
- Set acl-cache-size in xdbconfig.xml => default 32
- Keep #files in a folder < 200
- If # folders very large, keep #files in a folder > 100
- FTP: use binary mode when possible

Repository Views

- Analyze all indexes and tables involved
- Tune resource-view-cache-size to set cache size
- Equate under_path and equals_path with 1
  - Under_path(res,'/a/b/c') = 1

Oracle Text

Mohammad Faisal

Senior Development Manager

Query Performance

Relational Predicates and ORDER BY

- Problem: High number of Doc ID to ROWID resolutions ($R table)
- Solutions
  - Use FIRST_ROWS hint for ORDER BY SCORE() queries
  - Push resolution of equality predicates into Oracle Text index using sections
  - Push DATE range predicate into Oracle Text index
  - Use CTXCAT indextype instead of CONTEXT indextype for very short documents

Query Performance

Cache Hit

- Problem: Scattered I/O for Doc ID to ROWID resolutions
- Solution: Cache $R table for rowsource invocations or $K table for functional invocations of CONTAINS

- Problem: Poor cache hit ratio for Oracle Text index internal table blocks due to relatively low Oracle Text query load
- Solution: Use KEEP pool for all Oracle Text index internal tables e.g. $I, $R, $K

Query Performance

CONTAINS Query Re-write

- Problem: Multiple CONTAINS in the same SQL
- Solution: Merge into one CONTAINS (may have to use sections)

- Problem: Unnecessary use of expensive full-text operators
- Solution: Avoid wildcards, FUZZY, Stemming, Soundex

- Problem: Network and SQL overhead of multiple SQL
- Solution: Use Progressive Relaxation

Query Performance

SQL Plan

- Use FIRST_ROWS hint when using ORDER BY SCORE and selecting top-N hits using ROWNUM to optimize for response time. This will push the ORDER BY SCORE to Oracle Text.
- Do ANALYZE the Oracle Text index but NOT the Oracle Text index internal tables
- Make sure Oracle Text index access is not in the inner loop of NL

Query Performance

Full-text Operator Tuning

- Wildcard: Use Prefix Index or Substring Index
- FUZZY: Limit number of expansions
- WITHIN: Use Field sections or MDATA sections instead of Zone sections
- Stemming: Use Stem Index

Query Performance

Index Fragmentation and Garbage Collection

- Problem: Additional Reads of $I table due to logical fragmentation and garbage
- Solutions
  - Specify large amount of indexing memory for both CREATE INDEX and CTX_DDL.SYNC_INDEX
  - Use CTX_DDL.SYNC_INDEX infrequently to batch-up the workload
  - Use CTX_DDL.OPTIMIZE_INDEX frequently (I/O intensive)
    - If possible, avoid concurrent CTX_DDL.OPTIMIZE_INDEX and CTX_DDL.SYNC_INDEX
    - Use lightweight mode during business hours (e.g. TOKEN) and heavyweight mode otherwise (e.g. REBUILD)
      - Of all the heavyweight modes (FAST, FULL, REBUILD) the REBUILD mode provides the most reduction in additional Reads of $I table (but requires more temporary disk space)
        - Must apply patch for bug 5095220 before using REBUILD mode in parallel. Serial mode is OK without the patch.

Query Performance

Miscellaneous

- Transactional CONTAINS is expensive
- Avoid searching for very high frequency words by declaring them as Stop Words or modifying Lexer settings (e.g. by default e-mail address John.Smith@oracle.com is tokenized as JOHN SMITH ORACLE COM)
- If possible, partition the base table and create Oracle Text index as LOCAL (instead of GLOBAL), therefore reducing $I Read if the SQL query uses
  - partition elimination
  - ORDER BY partition column
- Reduce base table Reads by
  - Keeping the base table rows as narrow as possible, or
  - Creating a shadow table with minimal columns needed for CONTAINS, creating the Oracle Text index on the shadow table, and using USER_DATASTORE to index data from original base table

Indexing Performance

- If indexing binary documents, upgrade to patch-sets containing Verity filters instead of Stellent filters
  - 9iR2: 9.2.0.7 and later
  - 10gR1: 10.1.0.4 and later
  - 10gR2: all patch-sets
- Pre-filter binary documents (e.g. using CTX_DOC.POLICY_FILTER) and index filtered documents to avoid re-filtering during Document Service invocation
- In general, CREATE INDEX is several times faster than CTX_DDL.SYNC_INDEX
- If possible, avoid SYNC ON COMMIT due to base table contention for concurrent DML

Diagnostics
Timing

- In addition to general database timing tools, use CTX_OUTPUT package

Trace ID	Description
TRACE_IDX_USER_DATASTORE	Time spent executing user datastore
TRACE_IDX_AUTO_FILTER	Time spent invoking the AUTO_FILTER filter. (Replaces the deprecated TRACE_IDX_INSO_FILTER trace)
TRACE_QRY_XX_TIME	Time spent executing the $X cursor
TRACE_QRY_XF_TIME	Time spent fetching from $X
TRACE_QRY_IF_TIME	Time spent fetching the LOB locator from $I
TRACE_QRY_IR_TIME	Time spent reading $I LOB information
TRACE_QRY_R_TIME	Time spent fetching and reading $R information
TRACE_QRY_CON_TIME	Time spent in CONTAINS processing

Diagnostics
I/O

- In addition to general database timing tools, use CTX_OUTPUT package

Trace ID	Description
TRACE_QRY_X_ROWS	Total number of rows whose token metadata was fetched from $X
TRACE_QRY_I_ROWS	Number of rows whose $I token_info was actually read
TRACE_QRY_I_SIZE	Number of bytes read from $I LOBs

- In addition to general database I/O monitoring tools, use CTX_REPORT package to obtain Oracle Text index specific statistics, e.g. most frequent tokens, most fragmented tokens.

For More Information

http://search.oracle.com

Oracle Text 

or

http://www.oracle.com/technology/products/text/index.html

Panel

Bryn Llewellyn

Vishu Krishnamurthy

Nipun Agarwal/Mark Drake

Mohammad Faisal

Andrew Holdsworth

For More Information

http://search.oracle.com

or

http://www.oracle.com/


ORACLE