HERZLICH WILLKOMMEN

Oracle 11g Performance Tuning
Tipps und Tricks aus der Praxis

Autoren: Detlev Mörk (KDRS)

Axel Kraft (Trivadis)

Datum 18.07.2013

BASEL BERN LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MÜNCHEN STUTTGART WIEN

KDRS RZRS

trivadis makes IT easier.

AGENDA

1. Referenten
2. I/O und Disk Subsysteme
3. I/O-Durchsatzmessung mittels Calibration unter Oracle 11*g*
4. Tuning durch Function Based Index
5. MBRC und Workload/Systemstatistiken
6. Statistiken export, import, einfrieren.
7. Einsatz von Oracle Statspack.
8. Fragen ?

Referenten

- Detlev Mörk
  - Teamleiter Oracle Datenbanken
  - Zweckverband Kommunale Datenverarbeitung Region Stuttgart / Rechenzentrum Region Stuttgart GmbH
- Betrieb der Software für Gemeinden
  - Für z.B. Finanzverfahren, KFZ Zulassung, Standesamtswesen …
- Standorte
  - Stuttgart
  - Reutlingen
  - Ulm

Referenten

- Axel Kraft
  - Senior Consultant
  - Trivadis GmbH
- Infrastructure Managed Services
  - Oracle Database
- Standort
  - Stuttgart
- Oracle Erfahrung seit 1995

I/O und Disk Subsysteme - Die Praxis oder das Projekt

- Neue Verwaltungssoftware
  - Schalterverfahren
  - Über 500 Mandanten
  - Basis Oracle EE
  - Datenbank ca. 800GB
  - Applikation auf J2EE Basis
  - Active DataGuard für Auswertungssystem
  - Auswertesystem APEX Applikation

I/O und Disk Subsysteme - Die Umgebung

- IBM Server X3850
  - 2 CPU je 8 Cores
  - 256GB RAM
- SAN
  - IBM DS-8300
  - Netapp FAS3240
- OS
  - SLES 11 SP1 64-Bit
- DB
  - Oracle 11gR2 64-bit
- 3 Stufige Systeme Test, Konsolidierung, Produktion
  - Basissystem für unterschiedliche Verfahren (Datenbanken)

I/O und Disk Subsysteme - Vorbereitung

- Inbetriebnahme
  - Pilotbetrieb
  - Migration der restlichen Mandanten möglichst an einem (langen) WE

- Migrationstests
  - Vollmigration aller Mandanten mit 8 App Servern parallel.

  Resultat: zu langsam ☹

I/O und Disk Subsysteme - Wo liegt das Problem ?

- Beteiligte Systeme
  - Quell System DB2 zOS
    - Keine Last zu sehen
  - 8 App Server (Oracle Weblogic)
    - CPU Last ~15%
  - Oracle DB Server
    - CPU Last 5-10%
    - IO auf der Datenpartition 99% Util

I/O und Disk Subsysteme - „Use faster Disks“ im SAN

- Woher bekomme ich schnellere Disks?
- Sind die Disks langsam?

I/O und Disk Subsysteme - Auswertung während Migration

- Iostat
  - Iostat –xdm <device-id>

```
Device:         rrqm/s   wrqm/s     r/s     w/s    rMB/s    wMB/s avgrq-sz avgqu-sz   await  svctm  %util
dm-11             0.00     0.00  347.20  517.40     9.18     2.01    26.50     6.69    7.67   1.04  90.32

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.41    0.00    0.28    2.92    0.00   96.40

Device:         rrqm/s   wrqm/s     r/s     w/s    rMB/s    wMB/s avgrq-sz avgqu-sz   await  svctm  %util
dm-11             0.00     0.00  449.00  209.80    11.34     0.81    37.79     3.33    5.13   1.47  96.64

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.91    0.00    0.39    2.85    0.00   95.85
```

  - 9 - 11MB/s lesend bei 96% Auslastung

KDRS RZRS

trivadis makes IT easier.

I/O und Disk Subsysteme - Auswertung (2)

- Ist die Platte immer so langsam?
  - Test mit dd (iflag=direct → direct IO)
  - Blockgröße 8k (bs=8k) → 40,7MB/s lesend
  - Blockgröße 256k → 250MB/s , 1M → 336MB/s

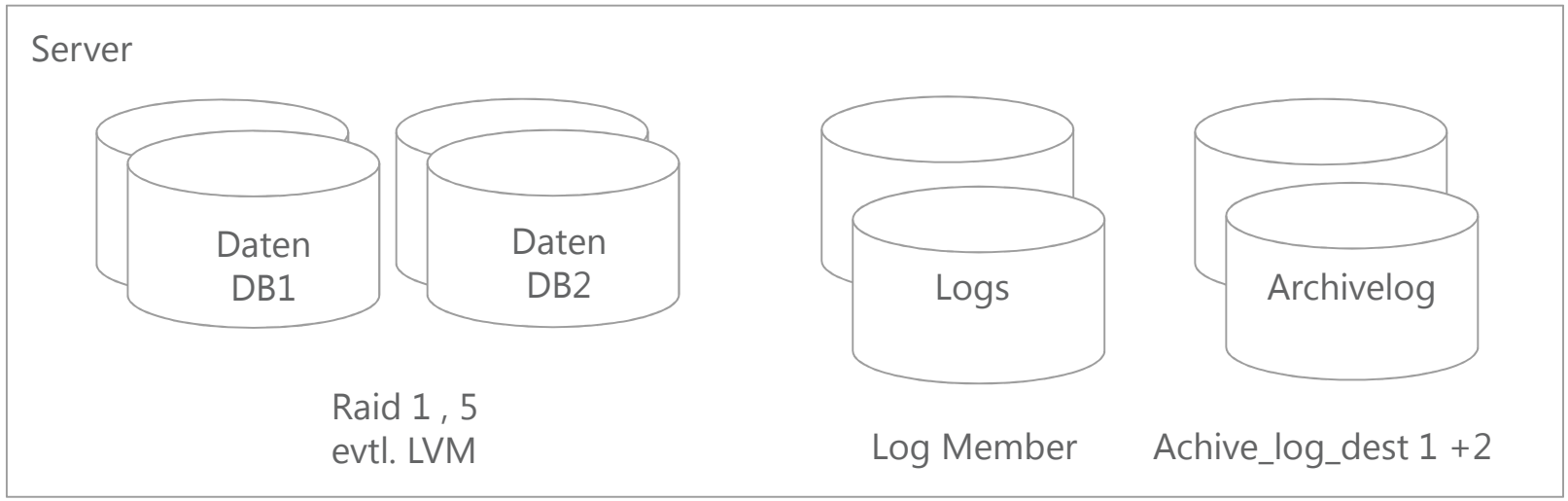
```
m1rrzsdbora12p:/home/oracle # dd of=/dev/null if=/dev/mapper/db026-db026 bs=8k count=256k iflag=direct
^C60311+0 records in
60310+0 records out
494059520 bytes (494 MB) copied, 12.1423 s, 40.7 MB/s

m1rrzsdbora12p:/home/oracle # dd of=/dev/null if=/dev/mapper/db026-db026 bs=256k count=256k iflag=direct
^C48775+0 records in
48774+0 records out
12785811456 bytes (13 GB) copied, 51.1101 s, 250 MB/s

m1rrzsdbora12p:/home/oracle # dd of=/dev/null if=/dev/mapper/db026-db026 bs=1024k count=256k iflag=direct
^C4608+0 records in
4607+0 records out
4830789632 bytes (4.8 GB) copied, 14.3648 s, 336 MB/s
```


KDRS RZRS

trivadis makes IT easier.

I/O und Disk Subsysteme - Oracle FS Layout

- Separieren von Tabellen und Indizes
- Separieren von Logfiles
- Separieren von Archive Logs

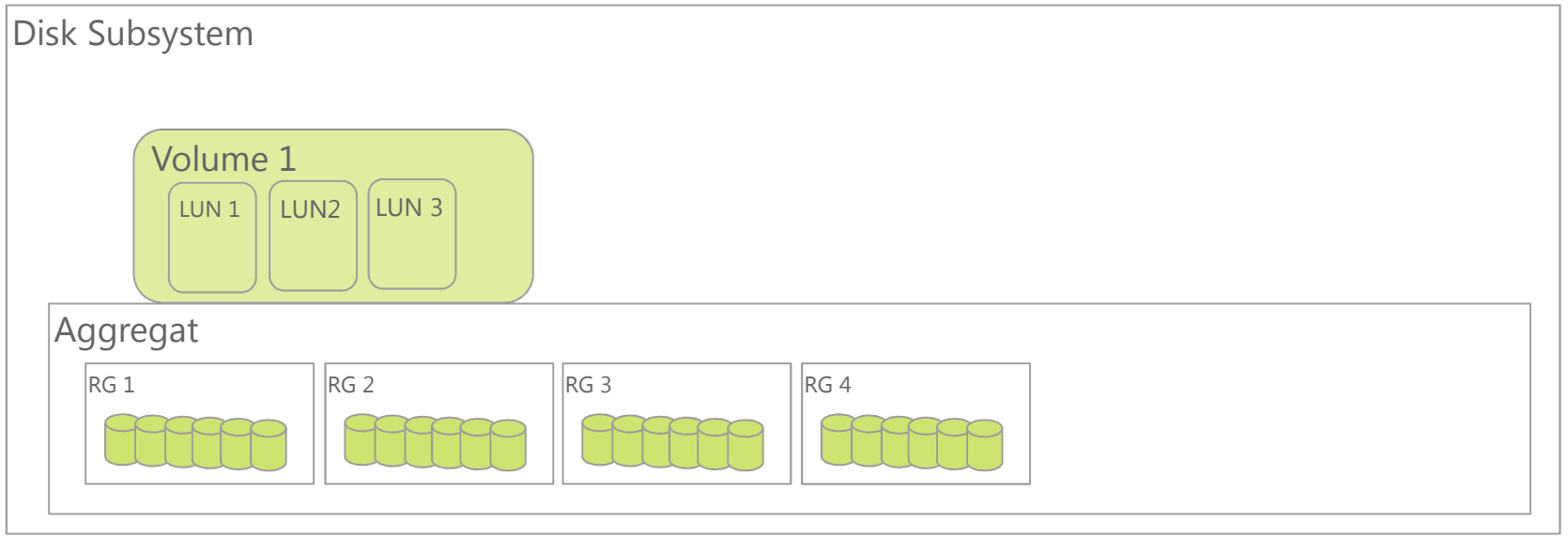


KDRS RZRS

trivadis makes IT easier.

I/O und Disk Subsysteme - Disk Subsystem

- Aufbau Disk Subsystem
  - Raidgruppen: Raid1, Raid 5, Raid DP
  - Stripesets aus Raidgruppen: Extentpool (IBM), Aggregat (Netapp)
  - IO Verteilung über alle Platten im Aggregat → viele IOPS
  - Volumes/LUNs

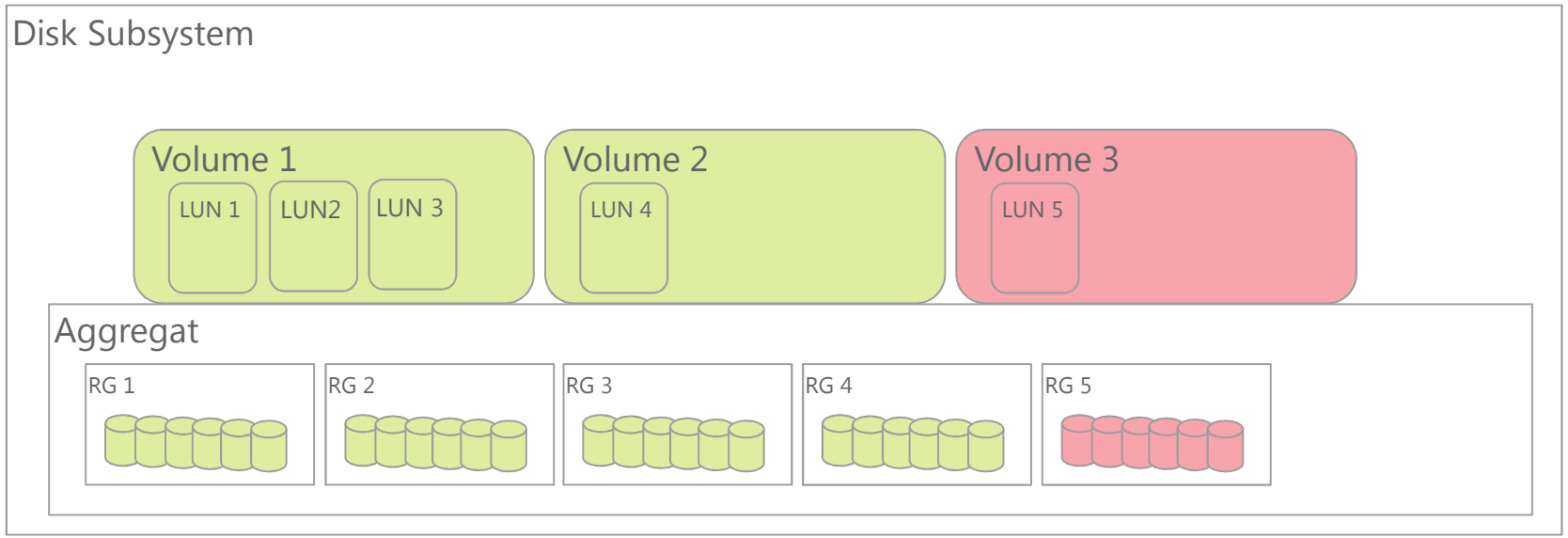


KDRS RZRS

trivadis makes IT easier.

I/O und Disk Subsysteme - Disk Subsystem

- Gefülltes Disk Subsystem
  - Rebalance?
    - Automatisch ☺
    - über Tools ☺
    - nicht möglich ohne zusätzlichen Platz ☹



I/O-Durchsatzmessung (Calibration_IO()) unter Oracle 11*g*

Erweiterung des Oracle Database Resource Managers.

- dbms_resource_manager.calibrate_io()
  - benutzt
    - db_block_size Größe für zufällige I/O's.
    - 1Mbyte Blocks für sequentielle I/O's.
- Realistische Messergebnisse, da Ausführung gegen Datenbankblocks erfolgt.
- Sollte bei geringer Datenbanklast ausgeführt werden.
- Voraussetzungen
  - Sysdba-Rechte
  - asynchrones I/O ist aktiviert

I/O-Durchsatzmessung (Calibration_IO()) unter Oracle 11*g*

- Parameter: FILESYSTEMIO_OPTIONS beachten.
  - Asynchronous I/O Support On Windows [ID 1228845.1]

```
SQL> select name,value
  2  from v$parameter
  3  where name in ('timed_statistics','filesystemio_options', 'disk_asynch_io')
  4  ;

NAME                           VALUE
------------------------------ ------------------
timed_statistics               TRUE
filesystemio_options           SETALL
disk_asynch_io                 TRUE
```

I/O-Durchsatzmessung (Calibration_IO()) unter Oracle 11*g*

- Datenfiles checken.

```
SQL> select d.name, f.asynch_io
  2  from v$datafile d inner join v$iostat_file f on d.file#=f.file_no;

NAME                                     ASYNCH_IO
---------------------------------------- ----------
/u01/oradata/DB001/system01.dbf          ASYNC_ON
/u01/oradata/DB001/system01.dbf          ASYNC_ON
/u01/oradata/DB001/sysaux01.dbf          ASYNC_ON
/u01/oradata/DB001/undotbs01.dbf         ASYNC_ON
/u01/oradata/DB001/users01.dbf           ASYNC_ON
/u01/oradata/DB001/example01.dbf         ASYNC_ON
/u01/oradata/DB001/perfstat01.dbf        ASYNC_ON
```

I/O-Durchsatzmessung (Calibration_IO()) unter Oracle 11*g*

- Calibrate_IO durchführen.

```
set serveroutput on
declare
  lat  INTEGER;
  iops INTEGER;
  mbps INTEGER;
begin
  dbms_resource_manager.calibrate_io (1, 10, iops, mbps, lat); -- P1: Disks, P2: max. Latenz
  dbms_output.put_line ('max_iops = ' || iops);
  dbms_output.put_line ('latency  = ' || lat);
  dbms_output.put_line ('max_mbps = ' || mbps);
end;
/
SQL>   2    3    4    5    6    7    8    9   10   11
max_iops = 79
latency  = 11
max_mbps = 32
```

Anmerkung: 1 HDD, SATA aus dem Disktop-Umfeld bringt etwa 80 IOPS zustande.

I/O-Durchsatzmessung (Calibration_IO()) unter Oracle 11*g*

- In der View v$io_calibration_status, den Runtime-Status während der Messung.
- Festhalten der Ergebnisse.
  - dba_rsrc_io_calibrate, als historisches Ergebnis

```
SQL> select start_time, end_time, max_iops, max_mbps, latency, num_physical_disks
  2  from dba_rsrc_io_calibrate;

START_TIME                END_TIME                  MAX_IOPS MAX_MBPS LATENCY NUM_PHYSICAL_DISKS
------------------------- ------------------------- -- ---------- ------ ------- ---------- ------------------------
31.05.13 11:06:50,197380 31.05.13 11:12:21,404812    79             32          11                          1
```

Tuning durch Function Based Index

- Verwenden bei Where Bedingungen mit z.B.
  - Upper
    - `Select * from person where upper(Vorname)= 'HANS';`
  - Lower
    - `Select * from person where lower(Nachname) = ,maier';`
  - Substr
    - `Select * from person where substr(geburtsdatum,1,4) = '1968';`
  - Concat
    - `Select * from person where Vorname||Nachname = 'HansMaier';`
  - eigene Pl/Sql Funktionen
    - `Select * from person where rueckwaerts('Hans') = 'snaH';`

Tuning durch Function Based Index

```
select * from WBNDRUCK wbn
    WHERE     wbn.ags = :1
       AND wbn.wvz_id = :2
       AND EXISTS(SELECT 1
                  FROM wbndruck wbn2
                 WHERE     wbn.ags = wbn2.ags
                   AND wbn.wvz_id = wbn2.wvz_id
                   AND wbn.adresse1|| wbn.adresse2
                    || wbn.adresse3|| wbn.adresse4
                    || wbn.adresse5|| wbn.adresse6
                    || wbn.adresse7|| wbn.adresse8
                    || wbn.adresse9 = wbn2.adresse1
                    || wbn2.adresse2 || wbn2.adresse3
                    || wbn2.adresse4 || wbn2.adresse5
                    || wbn2.adresse6 || wbn2.adresse7
                    || wbn2.adresse8 || wbn2.adresse9
                       AND wbn.barcode <> wbn2.barcode);
```


KDRS RZRS

trivadis makes IT easier.

Tuning durch Function Based Index

Zugriff ohne Function Based Index

Sql	Sharable Mem	Persistent Mem	Runtime Mem	Sorts	Executions	Parse Calls	Module	Buffer Gets Per E...	Buffer Gets	Disk Reads	Elaps...	Ver
UPDATE EWO.WBNDRUCK wbn SET wbn.WBN_MEHRWERTIG ...	48759	80960	79928	0	1	1	JDBC Thin Client	395523209	395523209	0	24m 39,7...	
[redacted]	49448	16272	15152	0	1	1	JDBC Thin Client	1606493	1606493	0	56,038588s	
	162066	61584	54960	0	29	0	JDBC Thin Client	6079	176277	606	2,899643s	
	138088	61400	53176	0	1	1	TOAD 11.6.1.6	109	109	0	0,026313s	
	134032	95992	87768	0	1	1	TOAD 11.6.1.6	144	144	0	0,015151s	
	69547	48432	40144	0	17	7	JDBC Thin Client	22	371	0	0,012898s	
	11591	1408	544	0	1	1		8	8	0	0,004051s	
	19816	4608	3648	0	3	3	Data Pump Worker	1	3	0	0,001184s	
	11563	1408	544	0	1	1		0	0	0	0,000842s	

SQL | Explain Plan | Sessions

Cached Explain Plan

Plan

- **UPDATE STATEMENT** ALL_ROWS
  Cost: 2
  - 6 **UPDATE** EWO.WBNDRUCK
    - 5 **NESTED LOOPS SEMI**
      Cost: 2 Bytes: 6 K Cardinality: 1
      - 2 **TABLE ACCESS BY INDEX ROWID TABLE** EWO.WBNDRUCK
        Cost: 1 Bytes: 3 K Cardinality: 1
        - 1 **INDEX RANGE SCAN INDEX (UNIQUE)** EWO.PK_WBNDRUCK
          Cost: 1 Cardinality: 1
      - 4 **TABLE ACCESS BY INDEX ROWID TABLE** EWO.WBNDRUCK
        Cost: 1 Bytes: 3 K Cardinality: 1
        - 3 **INDEX RANGE SCAN INDEX (UNIQUE)** EWO.PK_WBNDRUCK
          Cost: 1 Cardinality: 1

Tuning durch Function Based Index

Function Based Index anlegen

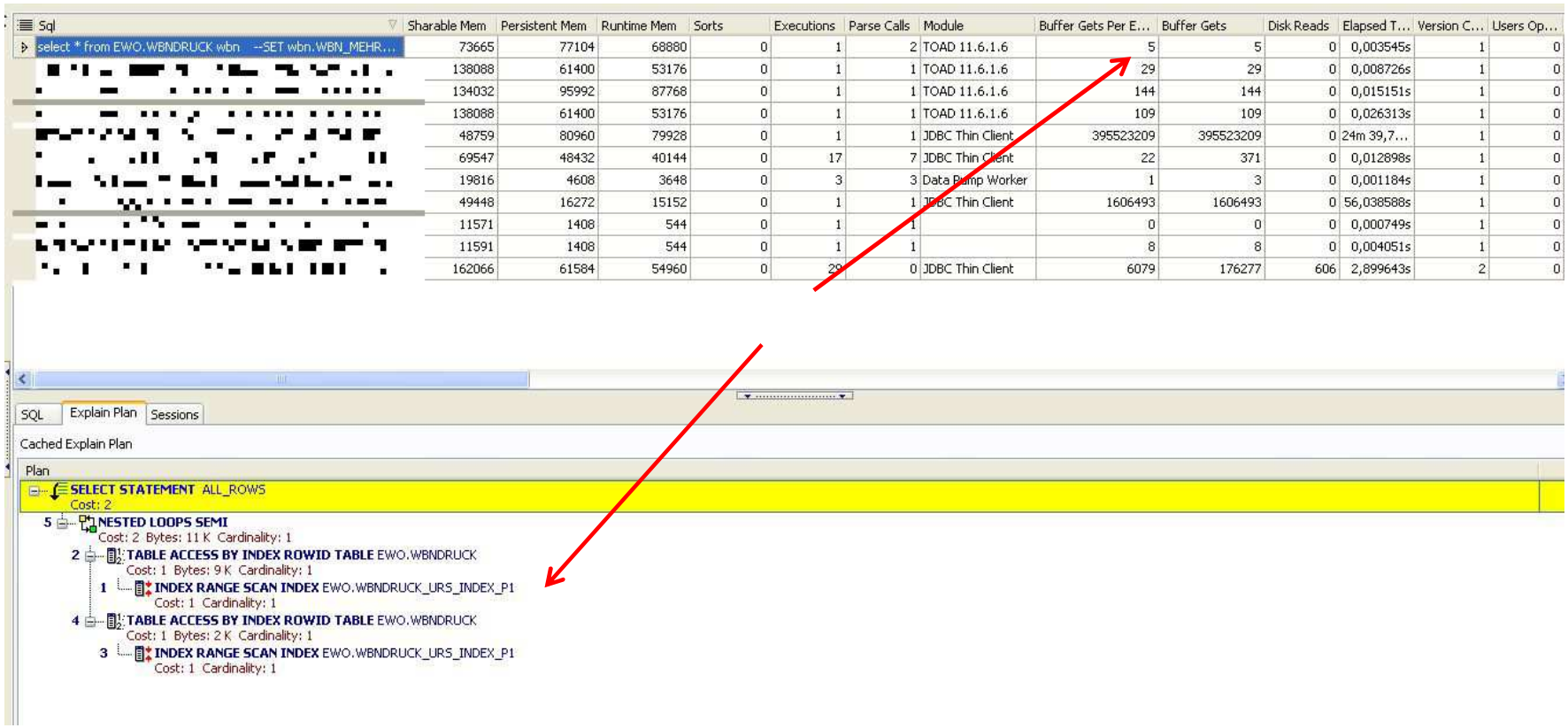
```
CREATE INDEX WBNDRUCK_URS_INDEX_P1 ON WBNDRUCK (AGS, WVZ_ID,
ADRESSE1"||"ADRESSE2"||"ADRESSE3"||"ADRESSE4"||"ADRESSE5"
||"ADRESSE6"||"ADRESSE7"||"ADRESSE8"||"ADRESSE9");
```

Analysieren der Tabellen

```
begin

  for rec in (select * from dba_tables where (table_name like '%WVZ%' or
                table_name like 'WBN%') and owner='XXX')
  loop
    dbms_output.put_line('Tabelle: '||rec.table_name);
    dbms_Stats.gather_table_Stats('XXX',rec.table_name ,cascade=>true);
  end loop;

end;
```

Tuning durch Function Based Index

Zugriff mit Function Based Index



Sql	Sharable Mem	Persistent Mem	Runtime Mem	Sorts	Executions	Parse Calls	Module	Buffer Gets Per E...	Buffer Gets	Disk Reads	Elapsed T...	Version C...	Users Op...
select * from EWO.WBNDRUCK wbn --SET wbn.WBN_MEHR...	73665	77104	68880	0	1	2	TOAD 11.6.1.6	5	5	0	0,003545s	1	0
[redacted]	138088	61400	53176	0	1	1	TOAD 11.6.1.6	29	29	0	0,008726s	1	0
	134032	95992	87768	0	1	1	TOAD 11.6.1.6	144	144	0	0,015151s	1	0
	138088	61400	53176	0	1	1	TOAD 11.6.1.6	109	109	0	0,026313s	1	0
	48759	80960	79928	0	1	1	JDBC Thin Client	395523209	395523209	0	24m 39,7...	1	0
	69547	48432	40144	0	17	7	JDBC Thin Client	22	371	0	0,012898s	1	0
	19816	4608	3648	0	3	3	Data Pump Worker	1	3	0	0,001184s	1	0
	49448	16272	15152	0	1	1	JDBC Thin Client	1606493	1606493	0	56,038588s	1	0
	11571	1408	544	0	1	1		0	0	0	0,000749s	1	0
	11591	1408	544	0	1	1		8	8	0	0,004051s	1	0
	162066	61584	54960	0	29	0	JDBC Thin Client	6079	176277	606	2,899643s	2	0

SQL | Explain Plan | Sessions

Cached Explain Plan

Plan

SELECT STATEMENT ALL_ROWS
Cost: 2
5 NESTED LOOPS SEMI
Cost: 2 Bytes: 11 K Cardinality: 1
2 TABLE ACCESS BY INDEX ROWID TABLE EWO.WBNDRUCK
Cost: 1 Bytes: 9 K Cardinality: 1
1 INDEX RANGE SCAN INDEX EWO.WBNDRUCK_URS_INDEX_P1
Cost: 1 Cardinality: 1
4 TABLE ACCESS BY INDEX ROWID TABLE EWO.WBNDRUCK
Cost: 1 Bytes: 2 K Cardinality: 1
3 INDEX RANGE SCAN INDEX EWO.WBNDRUCK_URS_INDEX_P1
Cost: 1 Cardinality: 1

KDRS RZRS

trivadis makes IT easier.

MBRC und Workload/Systemstatistiken

MBRC = Multi Block Read Count

- Anzahl Datenbankblöcke die bei einem Lesezugriff gelesen werden.
  - init.ora Parameter
    - db_file_multiblock_read_count

Workload/Systemstatistiken

- Workloadstatistiken stellen dem Optimizer Informationen zum I/O System zur Verfügung.
  - Sammeln der Werte mit:
    - exec dbms_stats.gather_system_stats(‚START‘);
    - exec dbms_stats.gather_system_stats(‚STOP‘);
    - Ergebnisse stehen in SYS.AUX_STATS$

MBRC und Workload/Systemstatistiken

Beispiel:

```
SQL> select pname name, pval1 value, pval2 info from sys.aux_stats$;

NAME                           VALUE INFO
------------------------------ ---------- ------------------------------
STATUS                            COMPLETED
DSTART                            09-18-2011 18:03
DSTOP                             09-18-2011 18:03
FLAGS                          1
CPUSPEEDNW                     1413.33874
IOSEEKTIM                        10 --Seek time + latency time + operating system
IOTFRSPEED                      4096 --Rate of a single read request in bytes/millisecond
...
```

Achtung:
Nach Ausführen von dbms_stats.gather_system_stats, sind die gesammelten Informationen sofort aktiv. Das sollte vermieden werden.
Der Optimizer verwendet ohne gesammlte Werte Default-Werte. (10ms Latenz, 4MB/s Durchsatz)

MBRC und Workload/Systemstatistiken

- Workload/Systemstatistiken sollten gesammelt und bewertet werden.
- Das Sammeln von Systemstatistiken ist aufwendig.
- Es muss ein Job eingerichtet werden, der in Intervallen die Daten sammelt, da NICHT jedes Mess-Intervall gültige Werte liefert.
- Nur wenn Werte für SREADTIM,MREADTIM,CPUSPEED und MBRC gesammelt werden konnten, liegen gültige Statistiken vor. Fehlt ein Wert dann ist das Set ungültig.
- Die Daten der Intervalle sind zu bewerten und Standardabweichungen zu berücksichtigen.
- Es ist sinnvoll CPU und I/O Werte unabhängig von Oracle zu sammeln und zu bewerten.
- Für die meisten SQL-Statments werden Workload/Systemstatistiken keine Auswirkung haben.

MBRC und Workload/Systemstatistiken

Beispiel:

```
begin
  dbms_scheduler.create_job( job_name => 'axk_workload',
                        job_type => 'plsql_block',
                        job_action => 'declare
                                        l_start varchar2(14); l_label varchar2(30);
                                      begin
                                        select to_char(sysdate, ''yyyymmddhh24mi'')
                                          into l_start from dual;
                                          l_label := ''Workload_'' || SYS_CONTEXT(''userenv'', ''instance_name'')|| ''_''|| l_start;
                                          dbms_stats.gather_system_stats( gathering_mode => ''INTERVAL'', interval => 10,
                                                                          statid => l_label, stattab => ''AXK_SYSTEM_WL'',
                                                                          statown => ''SYSTEM'');
                                       end;',
                        start_date => sysdate,
                        end_date => sysdate+35,
                        repeat_interval => 'FREQ=MINUTELY;INTERVAL=11' ,
                        enabled => true,
                        comments => 'AXK Gather Workload');
end;
/
```

MBRC und Workload/Systemstatistiken

- MBRC sollte nicht im Spfile gesetzt werden. Oracle bestimmt dann den Wert selber. Abhängig vom Betriebssystem. MOS [ID 841444.1]
  - Beispiel

    WAIT #140597730077512: nam='direct path read' ela= 18640 file number=15 first dba=521721

    block cnt=7 obj#=79148 tim=1358934683374796

    WAIT #140597730077512: nam='direct path read' ela= 599 file number=5 first dba=303746

    block cnt=126 obj#=79148 tim=1358934683467861

- Setzt man MBRC im SPFILE, Session oder Systemstatistiken, keine Dynamik mehr.
- Für die Kostenberechnung benutzt der Optimizer die 2 folgenden HIDDEN Parameter.
  - _db_file_optimizer_read_count – Kostenberechnung
  - _db_file_exec_read_count –I/O Request

Statistiken export, import, einfrieren

- Datenmigration in eine leere Datenbank
- Erste Mandanten schnell
  - Alle gehen schlafen
  - böses Erwachen am nächsten Morgen

  - Gesamtdauer nicht wie erwartet ☹
  - Wieder zu langsam

Statistiken export, import, einfrieren

- Was ist passiert?
  - Applikation liest in den angelegten Tabellen
  - Statistiken sagen, alle Tabellen sind leer
    - Optimizer entscheidet FTS bzw. Nested Loop ist eine gute Wahl ☹
- Was tun?
  - Optimizer auf den richtigen Weg bringen
    - Hint geht nicht, da die Applikation geliefert ist
- Lösung
  - Statistiken aus einer gefüllten Datenbank verwenden

Statistiken export, import, einfrieren

Statistiktabelle erzeugen

```
EXEC DBMS_STATS.create_stat_table(
       'KDRSNAC','XXX_STATS_TABLE');
-- Stat_schema, Stat_Tabelle
```

Statistik in Tabelle speichern

```
EXEC DBMS_STATS.export_schema_stats(
       'XXX','XXX_STATS_TABLE',NULL,'KDRSNAC');
-- Schema, Stat_Tabelle,Stat_id,Stat_schema
```

Statistiken laden (Nach Exp/Imp in andere DB)

```
EXEC DBMS_STATS.import_schema_stats(
<ZIEL_SCHEMA>,'XXX_STATS_TABLE',NULL,'KDRSNAC');
-- Schema, Stat_Tabelle, Stat_id, Stat_schema
```

Statistiken export, import, einfrieren

WICHTIG: Statistiken einfrieren (auto optimizer stats collection)

```
EXEC DBMS_STATS.lock_schema_stats('XXX');
```

sonst kommt wieder das böse Erwachen ☺

Statistiken können auch auf Tabellenebene eingefroren werden

```
begin

  for rec in (select * from dba_tables where (table_name like '%WVZ%'
                      or table_name like 'WBN%') and owner='XXX')
  loop
    dbms_stats.unlock_table_stats('XXX',rec.table_name);
    dbms_output.put_line('Tabelle: '||rec.table_name);
    dbms_Stats.gather_table_Stats('XXX',rec.table_name ,cascade=>true);
    dbms_stats.lock_table_stats('XXX',rec.table_name);
  end loop;

end;
```

Einsatz von Oracle Statspack

- Oracle Statspack = Oracle Statistik Package
- Einsatzzweck, Datenbankengpäße lokalisieren und zu einem späteren Zeitpunkt auswerten.
- Oracle Statspack ist ein kostenloses Tool seit Oracle 8.1.6 zur Sammlung/Auswertung von Datenbankinformationen.
- Diese Daten sind zur Performanceanalyse geeigent.
- Im Gegensatz zum kostenpflichtigen Automatic Workload Repository ist Oracle Statspack auch in der Oracle Standard Edition verfügbar. ☺
- Wichtig, AWR seit Oracle 10g verfügbar, ist in jeder installierten Datenbank verfügbar. Jedoch dürfen ohne Lizensierung des Diagnostic und Tuning Packs diese Daten nicht verwendet werden.
- In der Oracle Version 11g ist Oracle Statspack weiterhin kostenlos verfügbar.

Einsatz von Oracle Statspack

- Installation von Oracle Statspack
  - Annahme: Es existiert ein Tablespace mit dem Namen: PERFSTAT

```
SQL> @$ORACLE_HOME/rdbms/admin/spcreate.sql

Choose the PERFSTAT user's password
-----------------------------------
Not specifying a password will result in the installation FAILING

Enter value for perfstat_password: perfstat

Choose the Default tablespace for the PERFSTAT user
---------------------------------------------------
Below is the list of online tablespaces in this database which can
store user data.  Specifying the SYSTEM tablespace for the user's
default tablespace will result in the installation FAILING, as
using SYSTEM for performance data is not supported.

Choose the PERFSTAT users's default tablespace.  This is the tablespace
in which the STATSPACK tables and indexes will be created.

TABLESPACE_NAME                CONTENTS  STATSPACK DEFAULT TABLESPACE
------------------------------ --------- ----------------------------
EXAMPLE                        PERMANENT
PERFSTAT                       PERMANENT
.....................
```

Einsatz von Oracle Statspack

- Ersten Snapshot erstellen:

```
SQL> connect perfstat/<PWD>
Connected.
SQL> exec statspack.snap (i_snap_level => 7)
```

- Snap Level Default ist 5. Es werden alle Statistiken plus SQL-Statements gesammelt.
- Ab Level 6 werden Ausführungspläne im Repository gespeichert. Das ist empfehlenswert, da auch nun **historisch** auf Ausführungspläne zugegriffen werden kann. (Siehe Ausführungsplan mittels sprepsql.sql)

Einsatz von Oracle Statspack

- Im Level 7 werden zusätzlich Segment level statistics, inklusive Logical and Physical reads, Row Lock, ITL (Interested Transaction List) and Buffer Busy Waits gesammelt.

- Den Snap Level kann man auch permanent setzen:

```
SQL> connect perfstat/<PWD>
Connected.
SQL> exec statspack.snap(i_snap_level => 6, i_modify_parameter => 'true');
```

Einsatz von Oracle Statspack

- Oracle Statspack erstellt nicht automatisch weitere Snapshots.
- Deshalb einen Datenbank-Job für weitere Snapshots einrichten. Achtung Standard ist Level 5.

```
SQL> @$ORACLE_HOME/rdbms/admin/spauto.sql

.....
.....
SQL> select job, next_date, next_sec
  2    from user_jobs
  3   where job = :jobno;

       JOB NEXT_DATE         NEXT_SEC
---------- ----------------- --------------------------------
         3 09-JUL-13         14:00:00
.....
```

- Das Script spauto.sql richtet einen stündlichen Job ein. Um nicht mit dem AWR zu kollidieren sollte, der Job zu jeder halben Stunde laufen.

Einsatz von Oracle Statspack

- Erstellen eines Oracle Statspack Reports

```
SQL> @$ORACLE_HOME/rdbms/admin/spreport.sql

.....
Listing all Completed Snapshots
Instance     DB Name       Snap Id   Snap Started    Level Comment
------------ ------------ --------- ----------------- ----- --------------------
DGAXK        DGAXK              1 09 Jul 2013 13:40     7
                                2 09 Jul 2013 14:00     7

Specify the Begin and End Snapshot Ids
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter value for begin_snap: 1
Begin Snapshot Id specified: 1
.....
```

- Der Report wird als Text-File ausgegeben und kann mit einem Editor Programm angesehen werden.

Einsatz von Oracle Statspack

- House-Keeping von Oracle Statspack Snapshots.
  - Oracle liefert hier das Script: $ORACLE_HOME/rdbms/admin/sppurge.sql mit.
- Nachteil: Das Script prompted und verlangt Eingaben. So ist es nicht möglich einen automatischen Purge-Job einzurichten.
- Besser ist eine eigene PL/SQL Procedure zu erstellen, die die zu löschenden Snap IDs als Parameter entgegen nimmt.

```
begin
      snapshots_purged := statspack.purge( i_begin_snap      => <ERSTER_SNAPSHOT>
                                         , i_end_snap        => <ZWEITER_SNAPSHOT>
                                         , i_snap_range      => true
                                         , i_extended_purge  => false
                                         , i_dbid            => <DBID>
                                         , i_instance_number => <INSTANCE_NUMMER>);
       dbms_output.put_line('snapshots_purged: '||snapshots_purged);
end;
```

Einsatz von Oracle Statspack

- Ausführungsplan mittels sprepsql.sql ermittlen

```
 CPU             CPU per             Elapsed                                 Old
  Time (s)  Executions  Exec (s)  %Total   Time (s)    Buffer Gets  Hash Value
---------- ---------- ---------- ------ ---------- ---------------  ----------
.....
      1.36         12       0.11    4.0       4.19          22,591  4099805351
Module: SQL*Plus
insert into emp_copy select * from emp_copy
.....
```

SQL> @$**ORACLE_HOME/rdbms/admin/sprepsql.sql**

SQL> REM Zeitraum muss bekannt sein, wird ueber SnapId abgefragt.

Einsatz von Oracle Statspack

- SQL_ID von Oracle Statspack Old Hash Value ermitteln und direkte Abfrage im Shared Pool.

```
 CPU              CPU per          Elapsd                              Old
  Time (s)  Executions  Exec (s)  %Total   Time (s)    Buffer Gets  Hash Value
---------- ---------- ---------- ------ ---------- ---------------  ----------
.....
      1.36         12       0.11    4.0       4.19         22,591  4099805351
Module: SQL*Plus
insert into emp_copy select * from emp_copy
.....

SQL> select sql_id from stats$sql_summary where old_hash_value='4099805351';
SQL_ID
-------------
4a0qw2z0bpahd
```

Einsatz von Oracle Statspack

- Ausführungsplan des Statements im Shared Pool anzeigen.

```
SQL> select * from table(DBMS_XPLAN.DISPLAY_CURSOR('4a0qw2z0bpahd',0));

PLAN_TABLE_OUTPUT
------------------------------------
SQL_ID  4a0qw2z0bpahd, child number 0
------------------------------------
insert into emp_copy select * from emp_copy

Plan hash value: 281390045

-----------------------------------------------------------------------------------------------
| Id  | Operation                    | Name     | Rows  | Bytes | Cost (%CPU) | Time       |
-----------------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT             |          |       |       |    4 (100)  |            |
|   1 |  LOAD TABLE CONVENTIONAL     |          |       |       |             |            |

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------------------------
|   2 |   TABLE ACCESS FULL          | EMP_COPY |    56 |  4872 |    4   (0)  | 00:00:01   |
-----------------------------------------------------------------------------------------------
```

Fragen ?

BASEL BERN LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MÜNCHEN STUTTGART WIEN

KDRS RZRS

trivadis makes IT easier.

VIELEN DANK.

KDRS, Detlev Mörk

E-Mail: d.moerk@kdrs.de

Trivadis, Axel Kraft

E-Mail: axel.kraft@trivadis.com

BASEL BERN LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MÜNCHEN STUTTGART WIEN

KDRS RZRS

trivadis makes IT easier.