LOGIN

Alen Prodan

Sistemske statistike i kalibracija
I/O podsustava uz pomoć
Orion alata

software

Agenda

- Razvojni put Oracle optimizera
- Što znači trošak (COST) vrijednost ?

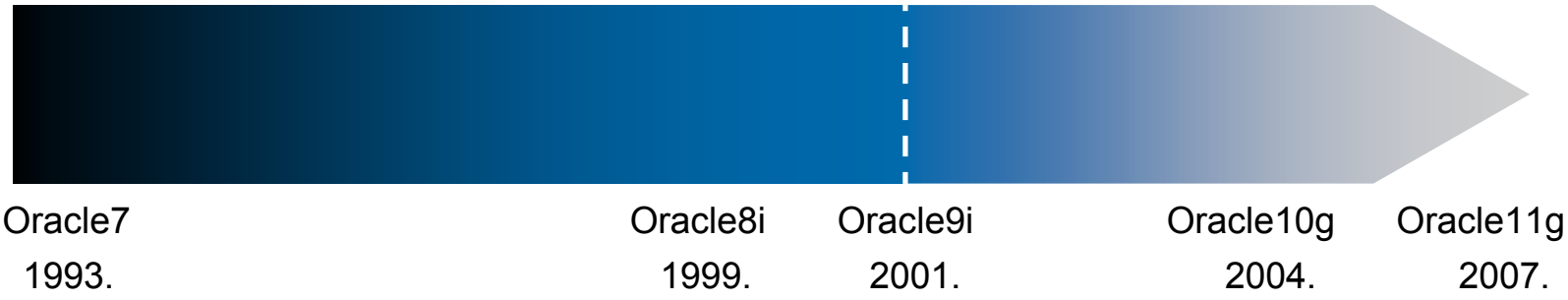
  Određivanje COST vrijednosti: usporedni prikaz tradicionalne metode i CPU costing modela (sistemskih statistika), pogreške u kalibraciji hardvera prilikom izračunavanja sistemskih statistika
- Upotreba Orion alata za kalibraciju hardvera

  Što je Orion alat, prednosti u korištenju alata, novosti u Oracle 11g RDBMS
- Zaključna riječ
- Pitanja i odgovori

Razvojni put Oracle optimizera

Razdoblje prije Oracle 9i RDBMS

- optimizer isključivo vodi računa o broju I/O operacija koje treba izvršiti za dohvat svih relevantnih podataka
- sustav ne uzima u obzir veličinu I/O operacija (singleblock I/O = multiblock I/O), niti brzinu njihova izvođenja
- U verziji Oracle 8.1 službeno se uvode optimizer_index_cost_adj i optimizer_index_caching parametri



Razvojni put Oracle optimizera

Razdoblje prije Oracle 9i

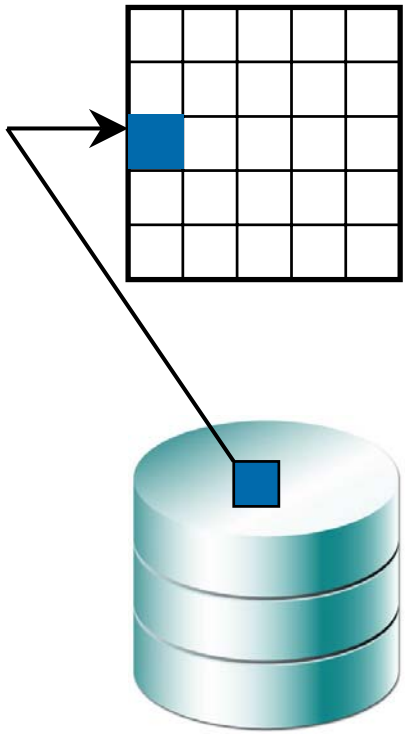
- optimizer_index_cost_adj utječe na ponašanje optimizera prilikom odabira indeks access pathova. Inicijalna postavka parametra iznosi 100, što znači da optimizer procjenjuje da je pristup putem indeksa podjednako zahtjevan kao i multiblock (full scan) operacija. Vrijednost parametra 50, označuje optimizeru da je pristup putem indeka upola "jeftiniji" od regularnog
- optimizer_index_caching - trošak izvršavanja putem indeksa ovisi o prisutnosti blokova tog indeksa u cache strukturama. Broj blokova u cacheu ovisi o faktorima koje CBO ne može predvidjeti, npr. opterećenje na sustavu i načinima pristupa različitih korisnika. Tim parametrom utječe se na pretpostavke optimizera vezano za prisutnost indeks blokova u cache strukturama. Inicijalna vrijednost parametra je 0.
- ti parametri imaju za cilj pružiti informacije optimizeru o tome koliko su singleblock I/O operacije "jeftiniije" od multiblock I/O operacija, dakle adresirati problem varijacija u tipu I/O operacija, kao i dati okvirnu ideju o zastupljenosti indeks blokova u cache strukturama

Razvojni put Oracle optimizera

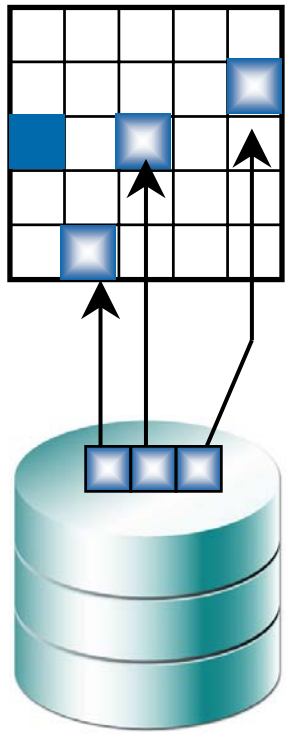
Oracle9i i novije verzije

- sistemske statistike, odnosno CPU costing model je po prvi puta zaživio u Oracle9i verziji RDBMS
- omogućuju cost-based optimizeru bolju procjenu potrebnih sistemskih resursa
- sistemske statistike se nameću kao idealno rješenje za "kalibraciju sustava", jer po prvi puta optimizer raspolaže ulaznim informacijama o učestalosti i vremenu trajanja pojedinih tipova I/O operacija, kao i o brzini CPU jedinice, a sve se temelji na stvarnom profilu hardvera Oracle poslužitelja

Prikaz singleblock i multiblock I/O mehanizma



singleblock I/O u jedan SGA buffer



multiblock I/O u više diskontinuiranih SGA buffera

Trošak (COST) vrijednost

Značenje pojma troška (cost)

- trošak (COST) predstavlja procjenu količine resursa potrebne za izvođenje neke operacije
- optimizer nastoji osigurati što efikasnije korištenje disk I/O, CPU i memorijskih resursa
- u konačnici, trošak predstavlja procjenu broja disk I/O operacija, te količinu CPU vremena i memorijskih resursa korištenih prilikom izvođenja SQL operacija
- operacija može biti skeniranje tablice, pristupanje slogovima u tablici koristeći indeks, join operacije, sort operacije i sl.
- pod troškom (cost) istovremeno podrazumijevamo osim količine utrošenih resursa i vrijednost dodjeljenu svakoj pojedinoj operaciji u sklopu plana izvršavanja

Trošak (COST) vrijednost

Značenje pojma troška (cost)

- **suprotno očekivanju, moguća je pojava kada manja cost vrijednost plana izvršavanja ne znači nužno i efikasnije (brže) izvođenje !**
- Trošak (cost) predstavlja procjenu vremena izvršavanja SQL naredbe, generiranu od strane modela ugrađenog u RDBMS softver, pa je samim time i podložan sljedećim greškama:
  - krive pretpostavke ugrađene u model
  - statistike o distribuciji podataka postoje, ali ne odražavaju trenutno činjenično stanje
  - statistike o distribuciji podataka ne postoje
  - nedostupni podaci o karakteristikama performansi hardvera na sustavu
  - mogući bugovi u modelu

Određivanje troška (cost) vrijednosti

Metode određivanja COST vrijednosti:

- **tradicionalna metoda**: temelji se isključivo na broju I/O operacija
- **sistemske statistike**: uzima u obzir veličinu i vrijeme odziva potrebno za I/O operacije

Opis postupka izvođenja testova

Konfiguriranje testnog okruženja

- svi testovi izvedeni su na Oracle 10g 10.1.0.5 Enteprise Edition na Windows 32bit platformi
- testovi su izvedeni u sljedećoj okolini kako bi se omogućilo reproduciranje rezultata u istovjetnim uvjetima:
  - db_block_size = 8 KB
  - db_file_multiblock_read_count = 8
  - upotreba locally managed tablespacea
  - veličina ekstenata 1 MB
  - manualni freelist space management (bez ASSM)
  - optimizer mode = ALL_ROWS
  - isključene sistemske statistike (u početnoj fazi)

Opis postupka izvođenja testova

kreiranje tablespacea

```
create tablespace hroug_tbs
datafile
'c:\oracle\oradata\ora101\hroug_tbs01.dbf' size 262208 k
extent management local
uniform size 1m
segment space management manual;
```

kreiranje testne tablice - svaki redak mora biti u zasebnom bloku

```
CREATE TABLE t ( id number, a char(2000), b char(2000));
INSERT INTO t
SELECT rownum, rownum, rownum
FROM all_objects
WHERE rownum <= 10000;
COMMIT;
```

prikupljanje CBO statistika

```
BEGIN
DBMS_STATS.GATHER_TABLE_STATS(
ownname=>user,
tabname=>'T',
estimate_percent=>DBMS_STATS.AUTO_SAMPLE_SIZE,
method_opt=>'FOR ALL COLUMNS SIZE AUTO');
END;
```

Tradicionalna metoda određivanja troška (cost)

- pojavljuje se u Oracle7 verziji i od tada se kontinuirano usavršava
- temelji se isključivo na broju I/O operacija, što predstavlja ujedno i njen najveći nedostatak
- nedostaci se prije svega odnose na vrednovanje utjecaja cache struktura i varijabilnosti troška I/O operacija, a ublaženi su u Oracle8i verziji uvođenjem optimizer_index_caching i optimizer_index_cost_adj parametara
- kod full scan operacija ključnu ulogu ima db_file_multiblock_read_count inicijalizacijski parametar

Tradicionalna metoda određivanja troška (cost)

db_file_multiblock_read_count parametar

- definira maksimalni broj blokova koje Oracle serverski proces može pročitati u jednoj multiblock I/O operaciji
- veća vrijednost parametra u pravilu znači više pročitanih blokova u jednoj operaciji, odnosno efikasnije izvođenje, a to pravilo vrijedi dok se ne dostigne točka opadanja performansi, gdje daljnje povećanje vrijednosti parametra može dovesti i do pogoršanja performansi sustava
- maksimalna vrijednost parametra zavisna je o platformi na kojoj se izvršava Oracle server
- na broj pročitanih blokova kod multiblock operacija djeluju sljedeća ograničenja:
  - maksimalni broj pročitanih blokova ovisan je o fizičkom ograničenju operativnog sustava, odnosno maksimalnoj fizičkoj veličini I/O
  - multiblock I/O operacija ne može čitati više blokova odjednom ukoliko oni prelaze granice ekstenata
  - multiblock I/O se može razlomiti i na više manjih operacija ukoliko se neki od blokova već nalaze u buffer cache strukturi

Tradicionalna metoda određivanja troška (cost)

db_file_multiblock_read_count parametar

- multiblock I/O operacije možemo pratiti uz pomoć SQL trace datoteka, a vidljive su kao db file scattered read wait eventi

```
PARSING IN CURSOR #4 len=21 dep=0 uid=70 oct=3 lid=70 tim=4586095223
hv=3395422149
ad='6cf903f0'
select max(id)
from t
END OF STMT
PARSE #4:c=15625,e=50764,p=1,cr=3,cu=0,mis=1,r=0,dep=0,og=1,tim=4586095216
BINDS #4:
EXEC #4:c=0,e=629,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,tim=4586099457
WAIT #4: nam='SQL*Net message to client' ela= 4 p1=1111838976 p2=1 p3=0
WAIT #4: nam='db file scattered read' ela= 35439 p1=6 p2=10 p3=8
WAIT #4: nam='db file scattered read' ela= 928 p1=6 p2=18 p3=8
WAIT #4: nam='db file scattered read' ela= 1209 p1=6 p2=26 p3=8
....
WAIT #4: nam='db file scattered read' ela= 1160 p1=6 p2=114 p3=8
WAIT #4: nam='db file scattered read' ela= 1104 p1=6 p2=122 p3=8
WAIT #4: nam='db file scattered read' ela= 2141 p1=6 p2=130 p3=7
```

Tradicionalna metoda određivanja troška (cost)

- upoznati sa ulogom db_file_multiblock_read_count parametra, možemo lakše sagledati njegov utjecaj na model određivanja cost vrijednosti fullscan operacija

```
SQL> set autot traceonly explain
SQL>
SQL> select max(id)
  2 from t;
Execution Plan
----------------------------------------------------------
    0   SELECT STATEMENT Optimizer=ALL_ROWS (Cost=1520 Card=1 Bytes=4)
    1 0   SORT (AGGREGATE)
    2 1     TABLE ACCESS (FULL) OF 'T' (TABLE) (Cost=1520 Card=10000 Bytes=40000)
```

- s obzirom da ne postoje indeksi nad testnom tablicom, jedini izbor optimizeru je tablescan operacija
- iz outputa autotrace komande moguće je iščitati 3 kritične vrijednosti na većini redaka plana izvršavanja:
  - trošak za pojedini redak (Cost)
  - procijenjeni broj redaka na izlazu iz te faze plana izvršavanja (Card)
  - procjena volumena podataka generiranih u tom koraku izvršavanja (Bytes)

Tradicionalna metoda određivanja troška (cost)

```
SQL> set autot traceonly explain
SQL>
SQL> select max(id)
  2 from t;
Execution Plan
----------------------------------------------------------
    0   SELECT STATEMENT Optimizer=ALL_ROWS (Cost=1520 Card=1 Bytes=4)
    1 0   SORT (AGGREGATE)
    2 1     TABLE ACCESS (FULL) OF 'T' (TABLE) (Cost=1520 Card=10000 Bytes=40000)
```

- u ovom slučaju SELECT naredba vraća jedan redak veličine 4 bytea, sa troškom Cost=1520.
- iznos od 1520 predstavlja zapravo trošak tablescan operacije iz retka 2, koji prosljeđuje 10.000 redaka i volumen od 40.000 byteova retku 1, odnosno SORT operaciji kao rezultat max() funkcije
- u retku 0 vidljiv je ukupan trošak za cijelu SQL naredbu, broj redaka kojeg vraća SQL naredba klijentu, te volumen podataka

Tradicionalna metoda određivanja troška (cost)

Utjecaj db_file_multiblock_read_count parametra na vrijednost troška

db_file_multiblock_ read_count	broj blokova	COST	(broj blokova/COST) korigirani dbfmbrc
4	10003	2397	10003 / 2397 = **4,17**
8	10003	1519	10003 / 1519 = **6,59**
16	10003	963	10003 / 963 = **10,39**
32	10003	610	10003 / 610 = **16,40**
64	10003	387	10003 / 387 = **25,85**
128	10003	245	10003 / 245 = **40,83**
256	10003	245	10003 / 245 = **40,83**

- u prethodnoj tablici vidljivo je variranje cost vrijednosti ovisno o promjeni db_file_multiblock_read_count parametra - što je veća vrijednost *dbfmbrc* parametra, to je cost vrijednost manja
- vrijednosti iz 4. stupca tablice dobivene su tako što je broj blokova iz 2. stupca podijeljen sa cost vrijednosti iz 3. stupca
- korigirani *dbfmbrc* iz stupca 4. mnogo sporije raste u odnosu na *dbfmbrc* iz stupca 1., zbog toga što on predstavlja korigiranu vrijednost koju Oracle automatski korigira ugrađujući u nju dozu pesimizma

Tradicionalna metoda određivanja troška (cost)

Utjecaj ugrađenog pesimizma na korigiranu vrijednost *dbfmbrc*

db_file_multiblock_ read_count	broj blokova	COST	(broj blokova/COST)
4	10003	2397	10003 / 2397 = **4,17**
8	10003	1519	10003 / 1519 = **6,59**
16	10003	963	10003 / 963 = **10,39**
32	10003	610	10003 / 610 = **16,40**
64	10003	387	10003 / 387 = **25,85**
128	10003	245	10003 / 245 = **40,83**
256	10003	245	10003 / 245 = **40,83**

- velike db_file_multiblock_read_count vrijednosti mogu izazvati neprirodno nagli pad COST vrijednosti (prividna efikasnost FTS)
- utjecaj tehničkih ograničenja na izvođenje velikih multiblock operacija:
  - efekt jednog takvog ograničenja je vidljiv u tablici u retku za dbfmbrc=256, gdje je cost vrijednost ostala nepromijenjena iz razloga što je 128*8KB=1024 KB maksimalna fizička veličina I/O operacije koju podržava operativni sustav, a Oracle raspolaže tom informacijom

Tradicionalna metoda određivanja troška (cost)

Uloga korigiranog *dbfmbrc* u izračunu cost vrijednosti

- korigirani *dbfmbrc* predstavlja ključ za izračunavanje cost vrijednosti full scan operacija u tradicionalnom modelu izračuna cost vrijednosti
- Npr., pretpostavimo da želimo izračunati cost vrijednost SQL naredbe koja mora scanirati objekt od 250.000 blokova, a sa postavkom db_file_multiblock_read_count=16 !

  U tom je slučaju korigirani dbfmbrc = 10,39.

Sa nevedenim ulaznim parametrima cost vrijednost iznosi:

COST = broj blokova / korigirani dbfmbrc

COST = 250.000 / 10,398 = 24043

Tradicionalna metoda određivanja troška (cost)

Uloga korigiranog *dbfmbrc* u izračunu cost vrijednosti

COST = broj blokova / korigirani dbfmbrc

COST = 250.000 / 10,398 = 24043

Primjerom možemo dokazati točnost izračuna:

```
CREATE TABLE t_250000
( id number );
exec DBMS_STATS.SET_TABLE_STATS(user, 'T_250000', numblks=>250000);
alter session set db_file_multiblock_read_count = 16;
```

```
set autot traceonly explain
SELECT max(id) FROM t_250000;
Execution Plan
----------------------------------------------------------
   0    SELECT STATEMENT Optimizer=ALL_ROWS (Cost=24044 Card=1 Bytes=13)
   1  0   SORT (AGGREGATE)
   2  1     TABLE ACCESS (FULL) OF 'T_250000' (TABLE) (Cost=24044 Card=2000
Bytes=26000)
```

Tradicionalna metoda određivanja troška (cost)

_table_scan_cost_plus_one - reguliranje fts/indeks access pathova kod SQL sa malom COST vrijednosti

```
set autot traceonly explain
SELECT max(id) FROM t_250000;
Execution Plan
----------------------------------------------------------
   0    SELECT STATEMENT Optimizer=ALL_ROWS (Cost=24044 Card=1 Bytes=13)
   1  0   SORT (AGGREGATE)
   2  1     TABLE ACCESS (FULL) OF 'T_250000' (TABLE) (Cost=24044 Card=2000 Bytes=26000)
```

COST = 250.000 / 10,398 = 24043; AUTOTRACE = 24044

Odstupanje = 24044 - 24043 = 1

```
SQL> select a.ksppinm, b.ksppstvl, a.ksppdesc
2 from x$ksppi a, x$ksppcv b
3 where a.indx = b.indx
4* and a.ksppinm like '%table%scan%'
SQL> /
KSPPINM                   KSPPSTVL   KSPPDESC
------------------------- ---------- ---------------------------------------------
_table_scan_cost_plus_one TRUE       bump estimated full table scan and index ffs
                                     cost by one
SQL>
```

Sistemske statistike i CPU costing model

Pregled sistemskih statistika i CPU costing modela

- uvedene sa ciljem točnijeg određivanja troška singleblock vs. multiblock I/O operacija
- eleminira potrebu za traženjem balansa između fts-hash-join i index-nested loop-IN-LIST operacija ručnim podešavanjem optimizer_index_caching i optimizer_index_cost_adj parametara
- po prvi puta optimizer raspolaže sa informacijama o performansama i opterećenju diskovnog podsustava i CPU resursa Oracle servera
- prilikom formiranja plana izvršavanja, optimizer izračunava procjenu I/O i CPU troška
- u Oracle9i verziji sistemske su statistike inicijalno isključene, dok su kod Oracle10g automatski uključene (u noworkload načinu)
- postoje dva tipa sist. statistika: NOWORKLOAD i "glavne" sist. statistike
- dbms_stats.gather_system_stats('NOWORKLOAD');
- dbms_stats.gather_system_stats('START' / 'STOP');

Sistemske statistike i CPU costing model

Priprema testnog okruženja za CPU costing model

- za potrebe testiranja, sistemske statistike podešene su uz pomoć poziva DBMS_STATS.SET_SYSTEM_STATS procedure:

```
BEGIN
 DBMS_STATS.SET_SYSTEM_STATS('SREADTIM', '10');
 DBMS_STATS.SET_SYSTEM_STATS('MREADTIM', '20');
 DBMS_STATS.SET_SYSTEM_STATS('CPUSPEED', '1000');
 DBMS_STATS.SET_SYSTEM_STATS('MBRC', '5');
END;
```

- postavljene vrijednosti možemo provjeriti:

```
SQL> select sname, pname, pval1 from sys.aux_stats$;
SNAME                          PNAME                          PVAL1
------------------------------ ------------------------------ ----------
...
SYSSTATS_MAIN                  CPUSPEEDNW
SYSSTATS_MAIN                  IOSEEKTIM
SYSSTATS_MAIN                  IOTFRSPEED
SYSSTATS_MAIN                  SREADTIM                           10
SYSSTATS_MAIN                  MREADTIM                           20
SYSSTATS_MAIN                  CPUSPEED                         1000
SYSSTATS_MAIN                  MBRC                                5
SYSSTATS_MAIN                  MAXTHR
SYSSTATS_MAIN                  SLAVETHR
```

Sistemske statistike i CPU costing model

Opis parametara sistemskih statistika

Naziv parametra	Opis	Inicijalizacija	Osvježavanje ili podešavanje
cpuspeed	prosječan broj CPU ciklusa u sekundi	Prilikom startanja sust.	gather_system_stats (noworkld,interval,start|stop) ili ručno
ioseektim	ioseektim=(seek time+latency+OS overhead)	Prilikom startanja sust.	NOWORKLOAD stat.ili ručno
iotfrspeed	brzina kojom server čita podatke kod single block IO	Prilikom startanja sust.	NOWORKLOAD stat.ili ručno
maxthr	maksimalni protok podataka kroz IO podsustav	-	gather_system_stats (noworkld,interval,start|stop) ili ručno
slavethr	prosječni IO protok podataka kod parallel SQL	-	gather_system_stats (noworkld,interval,start|stop) ili ručno
sreadtim	prosječno vrijeme potrebno za random čitanje jednog bloka	-	gather_system_stats (noworkld,interval,start|stop) ili ručno
mreadtim	prosječno vrijeme potrebno za multiblock IO	-	gather_system_stats (noworkld,interval,start|stop) ili ručno
mbrc	prosječan broj pročitanih blokova u multiblock operacijama	-	gather_system_stats (noworkld,interval,start|stop) ili ručno

Sistemske statistike i CPU costing model

Određivanje cost vrijednosti uz pomoć stistemskih statistika

- Prema Oracle9i Database Performanse Tuning Guide and Reference (part.no. a96533-02) cost vrijednost u CPU costing modelu određuje se na sljedeći način:

```
Cost=( #SR*sreadtim + #MR*mreadtim + #CPUCycles/CPU speed) / sreadtim
```

- gdje su:

Cost - trošak operacije
#SR - broj singleblock I/O operacija
sreadtim - vrijeme potrebno za izvođenje jedne singleblock I/O operacije (uključuje vrijeme pozicioniranja (latencija) + transfer podataka)
#MR - broj multiblock I/O operacija
mreadtim - vrijeme potrebno za izvođenje jedne multiblock I/O operacije (uključuje vrijeme pozicioniranja (latencija) + transfer podataka)
CPUCycles - uključuje CPU vrijeme izvođenja upita + CPU vrijeme dohvata podataka(CPU trošak dohvata podataka iz buffer cachea)
cpuspeed - broj CPU ciklusa u sekundi

Sistemske statistike i CPU costing model

Određivanje cost vrijednosti uz pomoć stistemskih statistika

```
Cost=( #SR*sreadtim + #MR*mreadtim + #CPUCycles/CPU speed) / sreadtim
```

- iz prethodne formule vidljivo je da se cost vrijednost može raščlaniti na trošak I/O komponente (single + multiblock IO) i trošak CPU komponente (CPU cycles)
- ukupna cost vrijednost je i dalje svedena na broj singleblock operacija, iako bi direktno bilo moguće izraziti ju u vremenu potrebnom za izvršavanje upita
- to je napravljeno iz razloga kompatibilnosti sa prethodnim verzijama kako se ne bi bitno mijenjala cost vrijednost SQL operacija nakon migracije

Sistemske statistike i CPU costing model

Usporedba tradicionalne metode i CPU costing modela

```
SQL> set autot traceonly explain
SQL>
SQL> select /* TRADICIONALNA */ max(id)
  2 from t;
Execution Plan
----------------------------------------------------------
    0   SELECT STATEMENT Optimizer=ALL_ROWS (Cost=1520 Card=1 Bytes=4)
    1 0   SORT (AGGREGATE)
    2 1     TABLE ACCESS (FULL) OF 'T' (TABLE) (Cost=1520 Card=10000 Bytes=40000)
```

```
SQL> select /* SYS_CPU_MODEL */ max(id) from t;
Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=ALL_ROWS (Cost=4010 Card=1 Bytes=4)
   1 0      SORT (AGGREGATE)
   2 1        TABLE ACCESS (FULL) OF 'T' (TABLE) (Cost=4010 Card=10000 Bytes=40000)
```

- cost vrijednost porasla za više od 2.5 puta
- kod optimizer_index_* parametara, umjetno se smanjuje cost vrijednost indeks pristupa, bez korekcija cost vrijednosti FTS operacija, dok kod CPU costing modela sustav poskupljuje FTS operacije

Sistemske statistike i CPU costing model

CPU costing model - izračun **I/O komponente** troška

- elementi za izračun I/O komponente COST vrijednosti:

```
MBRC = 5
#SR = 0
SREADTIM = 10
user_tables.blocks = 10003
#MR = 10003 / 5 = 2000,6
MREADTIM = 20
```

```
Cost = ceil((#SR * sreadtim + #MR * mreadtim) / sreadtim)
     = ceil((0 * 10 + 2000,6 * 20) / 10) = ceil(0 + 40012 = 40012 / 10)
     = ceil(4001,2) = 4002
```

- s obzirom da je prikazana cost vrijednost u autotrace prikazu 4010, postoji razlika od 8 jedinica, koju možemo pripisati CPU komponenti + utjecaju *_table_scan_cost_plus_one* parametra

Sistemske statistike i CPU costing model

CPU costing model - izračun **CPU komponente** troška

- uz pomoć explain plan naredbe generiramo plan izvršavanja:

```
SQL> explain plan for
  2  select max(id) from t;
Explained.
```

- zatim, uz pomoć SQL upita dobijemo pohranjeni plan izvršavanja iz plan_tablice kao u sljedećem primjeru:

```
select 'Plan izvrsavanja (Oracle 10gR1 10.1.0.5 Win 32bit)' from dual
union all
select '----------------------------------------------------' from dual
union all
select
id||' '||parent_id||' '||rpad(' ', depth, ' ')||' '||operation||
nvl2(options, ' ('||options||') ', '')||
nvl2(object_name, ''''||object_name||'''', '')||
nvl2(cost, ' (Cost='||cost||' IO_cost='||io_cost||' CPU_cost='||cpu_cost||'
CPU_calc_cost='||round((cpu_cost/(1000*10000)), 0)||')', '')
from plan_table
connect by prior id = parent_id
start with id = 0
```

Sistemske statistike i CPU costing model

CPU costing model - izračun **CPU komponente** cost vrijednosti

- rezultat izvođenja SQL naredbe u skripti vidljiv je u sljedećem prikazu:

```
Plan izvršavanja (Oracle 10gR1 10.1.0.5 Win 32bit)
----------------------------------------------------
0   SELECT STATEMENT (Cost=4010 IO_cost=4003 CPU_cost=72535764 CPU_calc_cost=7)
1 0   SORT (AGGREGATE)
2 1    TABLE ACCESS (FULL) 'T' (Cost=4010 IO_cost=4003 CPU_cost=72535764 CPU_calc_cost=7)
```

- prema dobivenom rezultatu procijenjen broj CPU ciklusa iznosi 72535764, dok brzina CPU jedinice iznosi 1000 oracle operacija u sekundi. Uvrstimo li vrijednosti u formulu za izračun CPU komponente dobijemo:

```
CPU cost = round(CPU Cyles / (CPU speed * sreadtim))
         = round(72535764 / (1000 * 10)) = 7253,57
```

- zbog čega nastaje razlika ? - brzina CPU izražena je u mikrosekundama (MHz), dok su IO operacije izražene u milisekundama. Stoga ako milisekunde pretvorimo u mikrosekunde dobijemo da sreadtim izražen u milisekundama iznosi 10*1000 = 10000, pa slijedi:

```
CPU cost = round(72.535.764 / (1.000 * 10.000))
         = round(72.535.764 / 10.000.000)
         = round(7,2) = 7
```

Sistemske statistike i CPU costing model

CPU costing model - **I/O komponenta + CPU komponenta**

- sada kada raspolažemo sa obje komponente CPU costing modela dobijemo:

```
COST = 4002(IO) + 7(CPU) + 1(_table_scan_cost_plus_one) = 4010
```

- cost vrijednost iznosi 4010, ako znamo da taj broj označuje broj singleblock operacija, a sreadtim iznosi 10 milisekundi, onda ispada da je procjenjeno vrijeme izvršavanja SQL upita:

```
ceil(4010 operacija * 10 milisekundi / 1000) = ceil(40100 / 1000) = 41 sekunda
```

- u prilog tome ide i prikaz iz dbms_xplan alata:

```
SQL> explain plan for
2 select max(id) from t;
Explained.
SQL> select * from table(dbms_xplan.display);
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 2966233522
--------------------------------------------------------------------------------
| Id | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
| 0 | SELECT STATEMENT |      |     1 |     4 | 4010     (1)| 00:00:41 |
| 1 | SORT AGGREGATE   |      |     1 |     4 |            |          |
| 2 | TABLE ACCESS FULL| T    | 10000 | 40000 | 4010     (1)| 00:00:41 |
--------------------------------------------------------------------------------
```

Sistemske statistike i CPU costing model

Pogreške u kalibraciji hardvera prilikom generiranja sistemskih statistika

- ažurne vrijednosti sist. statistika koje odražavaju pravo stanje profila hardvera imaju kritični utjecaj na efikasnost CPU costing modela
- izbor vremenskog intervala unutar kojega se izvodi prikupljanje statistika od krucijalne je važnosti, jer mora predstavljati reprezentativno opterećenje sustava
- pogreške nastaju zbog sasvim jednostavnih uzroka kao npr. netipično opterećenje u trenutku prikupljanja, pa do kompleksnijih tehničkih uzroka, kao npr. velikog cachea na diskovnom sustavu koji uz pomoć prediktivnog read-ahead mehanizma stvara privid da su multiblock I/O operacije puno brže od singleblock I/O operacija
- kako izbjeći pogreške u mjerenju ?

Upotreba ORION alata za kalibraciju hardvera

Što je Orion alat ?

- Orion je alat za mjerenje performansi diskovnog podsustava poslužitelja baze podataka, bez potrebe da se uopće instalira Oracle RDBMS softver ili kreira baza podataka
- prednost u odnosu na druge alate slične namjene je u tome što je Orion iz temelja dizajniran za simulaciju tipičnih Oracle I/O operacija jer koristi isti softver stack koji upravlja I/O operacijama na "pravom" Oracle serveru
- podržani su sljedeći tipovi I/O operacija:
  - singleblock random I/O - tipično za OLTP okruženje
  - multiblock sekvencijalni I/O - tipično za DSS okruženje, backup operacije i sl.
  - multiblock random I/O - pattern izazvan većim brojem sekvencijalnih multiblock operacija
  - mješano opterećenje
- Orion može poslužiti za testiranje bilo kojeg diskovnog sustava koji podržava async IO uključujući: DAS, SAN i NAS uređaje.

Upotreba ORION alata za kalibraciju hardvera

Prednosti u korištenju Orion alata

- Orion alat pruža mogućnost testiranja različitih konfiguracija diskovnih sustava, prije nego što se uopće krene u instalaciju Oracle poslužitelja
- administratori baze podataka mogu koristiti Orion za vrednovanje i usporedbu različitih modela i konfiguracija diskovnih sustava, temeljeno na projiciranom opterećenju sustava u fazi produkcijskog korištenja
- posebno je zanimljiva mogućnost nezavisnog mjerenja performansi diskovnog podsustava bez interferencija, a izmjerene je veličine moguće pohraniti u Oracle data-dictionary uz pomoć dbms_stats.set_system_stats procedure, te time izbjeći opisane pogreške u kalibraciji hardvera sustava koji se javaljaju prilikom standardne procedure prikupljanja sistemskih statistika.

Upotreba ORION alata za kalibraciju hardvera

Simulacija uzastopnih FTS operacija i utjecaj na s.statistike

- za vrijeme trajanja intervala prikupljanja client je izvršavao uzastopno veći broj FTS operacija, sa sljedećim ishodom na rezultate mjerenja sistemskih statistika:

```
SQL> select sname, pname, pval1 from aux_stats$;

SNAME                          PNAME                          PVAL1
------------------------------ ------------------------------ ----------
...
SYSSTATS_MAIN                  CPUSPEEDNW                    1383.607
SYSSTATS_MAIN                  IOSEEKTIM                           10
SYSSTATS_MAIN                  IOTFRSPEED                        4096
SYSSTATS_MAIN                  SREADTIM                         7.447
SYSSTATS_MAIN                  MREADTIM                         3.091
SYSSTATS_MAIN                  CPUSPEED                          972
SYSSTATS_MAIN                  MBRC                                4
....
```

- sreadtim (**7.447**) > mreadtim (**3.091**) !!! - signalizira optimizeru da su multiblock operacije "jeftiniije" od singleblock operacija

Upotreba ORION alata za kalibraciju hardvera

Simulacija uzastopnih FTS operacija i utjecaj na sistemske statistike - mjerenje izvedeno uz pomoć Orion alata

- gather_system_stats = sreadtim (**7.447**) > mreadtim (**3.091**) !!!
- uz pomoć Orion alata izveden je test koristeći sljedeće parametre, a dobiveni rezulati prikazani su u donjoj tablici (output iz test1_summary.txt):

```
orion -run simple -testname test1 -num_disks 1 -verbose
```

Large/Small	1	2	3	4	5
0	15,27	29,55	42,47	53,38	63,59

- ORION = sreadtim (**15,27**)
- minimalna zabilježena latencija uz pomoć Orion alata za singleblock operacije iznosi 15,27 ms, što je veliko odstupanje u odnosu na 3.091 ms
- glavni uzrok tome je neprestano izvršavanje FTS, i neznatni broj singleblock operacija
- u sljedećem primjeru u PL/SQL bloku simulirati će se u nekoliko različitih sesija tipičan workload random singleblock operacija, te prikupiti sistemske statistike u tom intervalu

Upotreba ORION alata za kalibraciju hardvera

Simulacija random singleblock I/O uz pomoć PL/SQL bloka

- u više istovremenih sesija izveden je sljedeći PL/SQL kod:

```
declare
 type t is table of number;
 l_data t := t();
 l_char char(2000);
 l_rnd number;
begin
 select id bulk collect into l_data from t;
 for x in 1..&wlimit
 loop
  l_rnd := trunc(dbms_random.value(1, 10000));
  select a into l_char from t where id = l_data(l_rnd);
  dbms_lock.sleep(0.1);
 end loop;
end;
```

- Upitom iz aux_stats$ možemo potvrditi podudarnost sa Orion rezultatom:

```
SNAME                 PNAME                          PVAL1
------------ ------------------------------ --------- -------------
SYSSTATS_MAIN         SREADTIM                          14.262
SYSSTATS_MAIN         MREADTIM                            4.76
```

Oracle 11g - ugrađena podrška za I/O kalibraciju

DBMS_RESOURCE_MANAGER.CALIBRATE_IO

- I/O kalibracija pomaže u ocjenjivanju performansi diskovnog podsustava i određivanju da li je uzrok loših I/O performansi u diskovnom sustavu ili na razini baze podataka
- ugrađena I/O kalibracija pokreće random I/O operacije nad datotekama baze podataka a dobiveni rezultati bolje odražavaju pravu sliku stanja
- I/O kalibraciju pokrećemo uz pomoć dbms_resource_manager.calibrate_io procedure (primjer preuzet iz Performance Tuning Guide 11g (11.1)):

```
SET SERVEROUTPUT ON
DECLARE
   lat INTEGER;
   iops INTEGER;
   mbps INTEGER;
BEGIN
-- DBMS_RESOURCE_MANAGER.CALIBRATE_IO (<DISKS>, <MAX_LATENCY>, iops, mbps, lat);
   DBMS_RESOURCE_MANAGER.CALIBRATE_IO (2, 10, iops, mbps, lat);
   DBMS_OUTPUT.PUT_LINE ('max_iops = ' || iops);
   DBMS_OUTPUT.PUT_LINE ('latency = ' || lat);
   DBMS_OUTPUT.PUT_LINE ('max_mbps = ' || mbps);
END;

sys@ORA11.LOGIN.HR> /
max_iops = 113
latency = 7
max_mbps = 10
```

Zaključak

- zadnjih nekoliko inačica Oracle RDBMS karakterizira snažan razvoj cost based optimizera
- u Oracle9i po prvi su puta zaživjele sistemske statistike, odnosno uveden je CPU costing model
- to novo svojstvo omogućuje optimizeru bolju procjenu potrebnih sistemskih resursa (cost) za izvršavanje SQL upita
- sistemske statistike predstavljaju idealan alat za "kalibraciju sustava", jer pomoću njih optimizer raspolaže ulaznim informacijama o učestalosti i vremenu trajanja pojedinih tipova I/O operacija, te brzini CPU jedinice
- rutine za prikupljanje sistemskih statistika u idealnim uvjetima generiraju reprezentativne podatke koji ocrtavaju pravu sliku profila hardvera sustava
- za potvrdu reprezentativnosti dobivenih sistemskih statistika, kalibraciju I/O sustava moguće je izvesti uz pomoć Orion alata
- Orion alatom moguće je izvesti kalibraciju I/O sustava u izoliranom okruženju, bez potrebe za postojanjem baze podataka, generirajući tipično I/O opterećenje za Oracle server iz razloga što koristi isti I/O stack kao i punokrvni Oralce server

LOGIN

Pitanja i Odgovori