Grafisch worstelen met Statspack

Alternatief voor de Oracle interface

Oracle Statspack wordt al van 8.1.6 door Oracle als standaardvoorziening meegeleverd en is bedoeld als hulpgereedschap bij het tunen en het opsporen van de oorzaken van performanceproblemen. De installatie is en het schedulen van snapshots zijn vrij eenvoudig, maar de interface die door Oracle wordt aangeboden is niet erg gebruikersvriendelijk, en bovendien nog niet geïntegreerd in de IDE. Paul van Leeuwen beschrijft in dit artikel een eenvoudig maar krachtig alternatief daarvoor.

Op de add-on markt is een tool als Statspack Viewer beschikbaar, maar voor het geval zo'n oplossing niet in het ICT-beheerbudget past, is het hier beschreven alternatief eenvoudig, gratis en naar eigen behoefte zelf aan te passen en uit te breiden. Voor de grafieken gebruiken we Excel. Een alternatief zoals OpenOffice Calc is ook mogelijk, maar dat biedt helaas niet die mooie klik-en-klaar wizard voor een grafiekje. We gebruiken ook wat extra hulpmiddelen voor het schedulen van de schoningsacties op de statspackdata. Ook dat is met de standaardscripts wat omslachtig te realiseren. De sources kunnen van de website van Optimize: http://www.array.nl/opt/algemeen.htm worden gedownload.

Standaardrapport als uitgangspunt

Als uitgangspunt voor dit verhaal is het standaard statspack rapport (output van spreport.sql) gebruikt, met name de eerste pagina. In alle literatuur over statspack wordt steeds deze pagina genoemd als de belangrijkste, omdat op die ene pagina meteen alle belangrijke indicatoren zijn samengevat. De te onderscheiden categorieën zijn:

- Load Profile (delta's): hoe zwaar wordt de instance belast
- Instance efficiency percentages (delta's)
- Shared pool statistics
- Top 5 wait events (delta's)

Op shared pool statistics na zijn de bovenstaande indicatoren delta's. Met de term delta wordt bedoeld dat de verschilwaarden van de betreffende statistiekindicatoren zijn berekend uit de twee snapshottijdstippen waarop het rapport is gebaseerd. Met andere woorden, de meeste statspackdata zijn cumulatieven die optellen vanaf het moment dat de instance wordt opgestart. Vandaar dat spreport.sql waarschuwt wanneer de instance startupdata van de gekozen twee snapshot id's verschillen. De delta's hebben consequenties: bij het genereren van een statspack rapport dien je als DBA je er terdege van bewust te zijn dat de weergegeven statistieken slechts delta's zijn van het begin- en eindtijdstip. Tussenliggende snapshotdata dragen helemaal niets bij aan het rapport. De top-SQL is die van alleen het eindtijdstip, en dat wordt nergens vermeld! Samengevat biedt het standaard statspack rapport een leuke aanzet, maar het wordt nèt niet echt iets waar je erg veel aan hebt. Als je een rapport wilt draaien, is het eerste wat je je zult moeten afvragen: "Welke begin- en eindtijdstippen moet ik nemen?". In de praktijk zijn daar eigenlijk nauwelijks goede criteria voor aan te dragen. Een periode van 24 uur zou een goed ijkpunt kunnen zijn. Maar waarom eigenlijk precies? Wanneer je de top-SQL wilt bekijken dan is alleen dat laatste moment belangrijk. Verder is het vervelend dat een instance shutdown je rapport invalideert. Bedenk ook dat de statspack tabellen, nadat zij enige tijd regelmatig - doorgaans één keer per uur - snapshots gemaakt hebben, een aardige hoeveelheid historische data bevatten. Die historie is erg interessant. Als we daar nu overzichten - in tabelvorm maar uiteindelijk ook grafisch - van konden maken dan:

- zagen we meteen de pijnpunten waarop en wanneer de instance het zwaar had,
- konden we ook prognoses maken van bijvoorbeeld de belasting en de groei,
- konden we bepalen van welke begin- en eindsnapshot we een uitgebreid rapport wilden genereren.

Het maken van de delta's

Als je de historische ontwikkeling grafisch wilt weergeven zul je in de meeste gevallen delta's van de opeenvolgende snaps-

hots moeten maken. Het koppelen van opeenvolgende snapshot id's ligt dan voor de hand. Ter illustratie vindt u in onderstaand kader een iets vereenvoudigde versie van de selectie van de topwaits van alle snapshots:

```
select w2.snap_id
,      w2.event
,      (w2.total_waits - w1.total_waits)/
      ((s2.snap_time - s1.snap_time)*24)          "waits/hour"
,      (w2.time_waited - w1.time_waited)/
      ((s2.snap_time - s1.snap_time)*24)          "waittime(cs)/hour"
,      to_char(s2.snap_time,'MM-DD-YYYY HH24:MI:SS') snap_time
from perfstat.stats$system_event w1
,    perfstat.stats$system_event w2
,    perfstat.stats$snapshot     s1
,    perfstat.stats$snapshot     s2
where w1.event            = w2.event
and   s1.snap_id          = w1.snap_id
and   s1.dbid             = w1.dbid
and   s1.instance_number = w1.instance_number
and   s2.snap_id          = w2.snap_id
and   s2.dbid             = w2.dbid
and   s2.instance_number = w2.instance_number
and   s1.snap_id+1        = s2.snap_id
and   s1.dbid             = s2.dbid
and   s1.instance_number = s2.instance_number
and   s1.startup_time     = s2.startup_time
and   w1.event not in (select i.event
                       from   perfstat.stats$idle_event i
                      )
order by 3 desc, 4 desc
```

Dat lijkt op het eerste gezicht OK. Het performt ook niet eens slecht, maar het werkt slechts als de snapshot id's ook keurig op elkaar volgen - zonder gaten dus. In de praktijk valt dat helaas nog wel eens tegen. Oracle garandeert ook beslist niet dat sequences altijd zonder gaten gegenereerd worden. Zo kun je ongemerkt gemakkelijk de helft van je historische data overslaan. Zonde, maar nog erger is het wanneer meerdere instances aan één database gekoppeld zijn, zoals bij Oracle Parallel Server en Real Application Clusters, om nog maar te zwijgen over 10g. Al die instances schrijven hun snapshots in dezelfde tabellen. Opeenvolgende snapshot id's hebben dan ook helemaal geen betekenis meer. De daarvoor gevonden remedie is een compromis tussen de wens voor een redelijke performance en KISS (keep it straight and simple): een hulptabel. In de hulptabel worden de snapshot id's met het daaropvolgende snapshot id opgeslagen. Die hulptabel moet nu wel steeds na elk snapshot ververst worden. Een materialized view met on commit refresh is overigens wel geprobeerd, maar jammer dus. Hier volgt het genereren van de hulptabel sp_ss_chain (in-line views zijn daarbij onmisbaar):

```
create table SP_SS_CHAIN
as
SELECT ss1.snap_id
,      ss2.snap_id  next_snap_id
,      ss1.startup_time
FROM
( SELECT rownum snap_num
  ,    ss.*
  FROM ( SELECT *
         FROM PERFSTAT.STATS$SNAPSHOT
         ORDER BY instance_number, snap_id
       ) ss
) ss1,
( SELECT rownum snap_num
  ,    ss.*
  FROM ( SELECT *
         FROM PERFSTAT.STATS$SNAPSHOT
         ORDER BY instance_number, snap_id
       ) ss
) ss2
WHERE ss1.snap_num+1 = ss2.snap_num
AND   ss1.dbid = ss2.dbid
AND   ss1.instance_number = ss2.instance_number
AND   ss1.startup_time = ss2.startup_time
;
```

Het bijwerken na elke snapshot is bijna hetzelfde: truncate en insert as select. Daarmee wordt het ophalen van de topwaits nu als volgt aangepast:

```
select w2.snap_id
,      w2.event
,      (w2.total_waits - w1.total_waits)/
      ((s2.snap_time - s1.snap_time)*24)          "waits/hour"
,      (w2.time_waited - w1.time_waited)/
      ((s2.snap_time - s1.snap_time)*24)          "waittime(cs)/hour"
,      to_char(s2.snap_time,'MM-DD-YYYY HH24:MI:SS') snap_time
from perfstat.stats$system_event w1
,    perfstat.stats$system_event w2
,    perfstat.stats$snapshot s1
,    perfstat.stats$snapshot s2
,    sp_ss_chain c
where w1.event            = w2.event
and   s1.snap_id          = w1.snap_id
and   s1.dbid             = w1.dbid
and   s1.instance_number = w1.instance_number
and   s2.snap_id          = w2.snap_id
and   s2.dbid             = w2.dbid
and   s2.instance_number = w2.instance_number
and   s1.snap_id          = c.snap_id
and   s2.snap_id          = c.next_snap_id
and   s1.dbid             = s2.dbid
and   s1.instance_number = s2.instance_number
and   s1.startup_time     = s2.startup_time
and   w1.event not in (select i.event
                       from   perfstat.stats$idle_event i
                      )
order by 3 desc, 4 desc
```

Overigens is bovenstaand voorbeeld geënt op een 8i database. In een 9i database is de kolom time_waited (in centiseconden) in stats$system_event vervangen door time_waited_micro (in

microseconden). Dat scheelt een factor 10.000. Dat is meteen ook de enige aanpassing in de views die gemaakt moest worden bij implementatie in een 9i omgeving.

Op bovenstaand principe zijn een aantal views gebouwd en getest.
- Blockbufferratio
- Instance efficiency
- Load profile
- Shared pool statistics
- Top waits

De kolomnamen van de views zijn bewust gelijkgehouden aan de namen van de corresponderende variabelen in het oorspronkelijke statspack rapport spreport.sql. Op die manier kunnen wijzigingen in versies van statspack misschien wat gemakkelijker verwerkt worden. Met bovengenoemde views zijn de mogelijkheden beslist niet uitgeput. Denk bijvoorbeeld aan instance activity, tablespace en datafile IO, rollback segmenten, latches, dictionary cache, library cache en instance parameter wijzigingen. Allemaal zeer geschikt om in een rustig moment eens uit te werken. Het gaat hier nu in de eerste plaats om grafieken die mij op het spoor brengen van regelmatige verstoringen en opvallende onregelmatigheden in het gedrag van de instance.

Klik en klaar grafieken

Zoals gezegd - uiteindelijk willen we een grafiek zien. Als we nu de uitvoer van de query's op de views zodanig spoolen dat we netjes geformatteerde CSV-files (Comma Separated Values) genereren dan hebben we via Excel zó een leuke grafiek. We genereren eerst iets als onderstaand (uit de instance efficiency view met sp_inst_eff_csv.sql) met als extensie .csv:

```
"Snap ID","Snap time","Buffer Nowait %","Redo Nowait %","Buffer Hit %","In-memory
Sort %","Library Hit %","Soft Parse %","Execute to Parse %","Latch Hit %","Parse CPU /
Parse Elapsed %","% Non-Parse CPU",
2,"10-08-2003 15:55:51",100,100,98.28,98.4,94.1,96.28,-59.79,100,73.08,99.6,
3,"10-08-2003 16:13:03",100,100,98.08,97.62,99.42,98.28,-88.96,100,50,99.86,
4,"10-08-2003 17:12:04",100,100,98.78,98.97,98.9,98.9,-95.5,100,29.41,99.96,
5,"10-08-2003 18:12:04",100,100,98.79,99,99.91,99.77,-112.56,100,43.75,99.94,
.....
```

Dan dubbelklikken we vanuit de verkenner. We zien Excel-sheet openen, met alles in nette rijen en kolommen (zie tabel 1). Als alles in de eerste kolom staat dan dient de landinstelling (tijdelijk) te worden aangepast. Hier kom ik verderop in dit artikel op terug. Vervolgens selecteren we de kolom 'Snap time' en een aantal kolommen die op dat moment interessant zijn: hooguit twee of drie. Voor dit voorbeeld nemen we 'Library Hit %' en klikken op de grafiek wizard.

In stap 1 van de wizard kiezen we als type: lijn.
In stap 3 kiezen we op de 'Assen'-tab als primaire as 'Kategorie'
In stap 4 als nieuw werkblad met als naam 'Library Hit %'

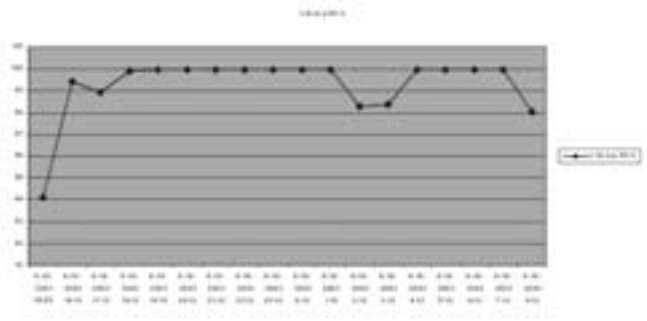
Na de laatste klik hebben we de grafiek (zie figuur 1).
Om dit soepel te laten verlopen dienen het gegenereerde CSV-formaat en de landinstelling onder Windows met elkaar te corresponderen. Het gaat met name om de datumnotatie en het veldscheidingsteken. In de scripts die van de Optimize site kunnen worden gedownload wordt uitgegaan van Amerikaans-Engels. Dat kan zonder veel problemen in de scripts sp_<category>_csv.sql worden aangepast. Vervang 'AMERICA" door 'THE NETHERLANDS' en de scheidingskomma's in de gegenereerde strings door punt-komma's.
Overigens is het tonen van grafieken nog eenvoudiger en mooier te realiseren als men een report web server heeft geïnstalleerd. Dit valt evenwel buiten het bestek van dit artikel.

Beheer

Als we elk uur een snapshot van de instance opslaan (op standaard niveau) is de ervaring dat dat op ongeveer 500 Mb per maand aan statspackdata uitkomt. Het door Oracle meegele-

Snap ID	Snap time	Buffer Nowait %	Redo Nowait %	Buffer Hit %	In-memory Sort %	Library Hit %
2	10/8/2003 15:55	100	100	98.28	98.4	94.1
3	10/8/2003 16:13	100	100	98.08	97.62	99.42
4	10/8/2003 17:12	100	100	98.78	98.97	98.9
5	10/8/2003 18:12	100	100	98.79	99	99.91
6	10/8/2003 19:12	100	100	98.78	98.97	99.95
7	10/8/2003 20:12	100	100	98.78	98.96	99.95
8	10/8/2003 21:12	100	100	98.78	98.97	99.95
9	10/8/2003 22:12	100	100	98.78	98.95	99.95
10	10/8/2003 23:12	100	100	98.78	98.97	99.95
11	10/9/2003 0:12	100	100	98.76	98.95	99.95

Tabel 1. De rijen en kolommen uit het beschreven Excel-sheet

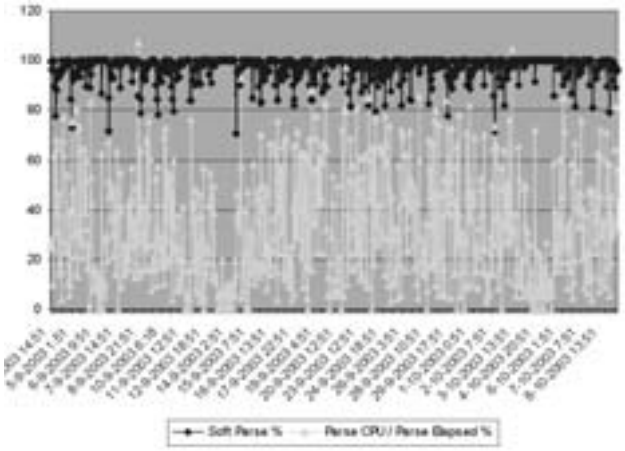


Figuur 1: De kolom 'Library Hit %' in grafiekvorm

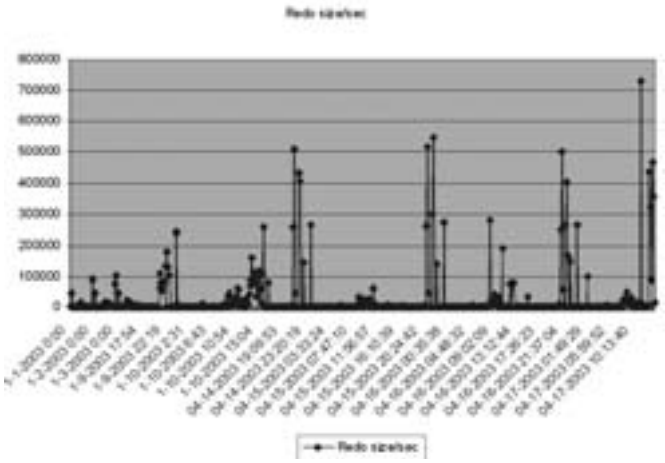
verde opschoningsscript is geënt op interactief gebruik. Ga dat maar eens elke week voor al je instances uitvoeren. En als we langer wachten (een ruime rollback/undo) is het noodzakelijk om - bij het ineens verwijderen van een maand of meer statspackdata - standaard purge script te gebruiken. Daarin wordt immers om begrijpelijke redenen niet gecommit. Als we niet alleen het maken van snapshots willen schedulen maar ook het opschonen dan ligt een package voor de hand. De package code zit ook bij de script die van de site gedownload kan worden. Deze package - spman - bevat procedures voor:

- Het maken van een snapshot inclusief het verversen van de hulptabel sp_ss_chain
- Het opschonen van snapshotdata met als parameter de bewaartijd in maanden inclusief verversen van sp_ss_chain.
- Het uitvoeren van snapshots op alle databases waarvoor een private database link zonder username/password voorhanden is inclusief verversen van sp_ss_chain.
- Het schonen van snapshotdata op alle databases waarvoor een private database link zonder username/password voorhanden is. De standaard bewaartijd is daarbij één maand.

Die twee laatste procedures zijn handig als je in de OEM niet voor elke instance een snap en een purge job wilt zien. Het verdient aanbeveling om alle views, de hulptabel en de package niet bij PERFSTAT maar in een eigen schema onder te brengen, bijvoorbeeld SPMONITOR. Deze gebruiker dient expliciet SELECT en DELETE rechten te hebben op de perfstat tabellen en EXECUTE rechten op de package statspack.



Figuur 2. Grafiek uit Instance Efficiency



Figuur 3. Grafiek uit Load Profile

Nawoord

Het ging erom Statspack toegankelijk te maken zonder dat daarvoor 'dure' oplossingen aangeschaft hoeven te worden. In de implementatie zijn we wel van de beschikbaarheid van MS Excel uitgegaan. Er zijn echter meer spreadsheetpakketten onder de zon. Ervaringen daarmee en tips daarover zijn van harte welkom bij de auteur. De bij dit artikel te downloaden scripts zijn oorspronkelijk geschreven en getest op Oracle 8.1.7. Voor 9i bleek slechts een minieme aanpassing in de top_waits view nodig, vanwege de overgang van centiseconden naar microseconden in v$system_event. Het is wel merkwaardig, dat dit in de documentatie van 9i voor deze dynamic performance view nog niet is bijgewerkt. Het meest interessant zijn uiteindelijk de opgeleverde grafieken. Figuur 2 en figuur 3 tonen een kleine selectie uit de resultaten.

Ir. Paul J. van Leeuwen MSc. als Oracle DBA & Consultant werkzaam bij CIBER Nederland (e-mail: paul.van.leeuwen@ciber..nl).