Timing features in Oracle 9i

Doorloop- en CPU-tijd nauwkeuriger bekijken

Voor monitoring, diagnose en herstel van performanceproblemen is timing-informatie onmisbaar. Wie op ratio's vaart, in plaats van op tijd-gebaseerde metingen, zal een volstrekt verkeerde richting uit worden gestuurd. Oracle9i Release 1 bevat verschillende veranderingen en uitbreidingen die het mogelijk maken om accurate timinginformatie te verzamelen. Naast veranderingen in de kernel van Oracle9i, zijn er in bepaalde v$-views interessante kolommen die nog niet in brede kring bekend zijn, laat staan gebruikt worden. In dit artikel bespreken we een aantal van deze veranderingen en laten we zien hoe ze door databasebeheerders kunnen worden gebruikt. We zullen ook duidelijk maken welke onderdelen de Oraclekernel op dit moment nog altijd niet zichtbaar weet te maken.

Nauwkeurigheid van de tijdsinformatie

Onderstaande tabel laat zien wat de meest gebruikte nauwkeurigheden zijn die op dit moment voor tijdsinformatie worden gehanteerd.

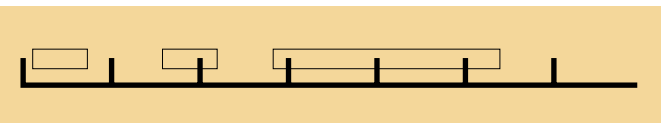
NAUWKEURIGHEID	TIMING
Seconde	1
Centiseconde	1/100
Milliseconde	1/1000
Microseconde	1/1.000.000
nanoseconde	1/1.000.000.000

Tabel 1. Timingnauwkeurigheid

Timingproblemen

Voor het timen van bewerkingen maakte de Oracle-kernel jarenlang gebruik van centiseconden. Het fundamentele probleem van het timen in centiseconden is dat, als een bepaalde bewerking minder dan een centiseconde in beslag neemt (bijvoorbeeld 600.000 microsconden), hij geteld wordt als 0.

Hoewel op het eerste gezicht een voor de hand liggende oplossing, gaat hij er van uit dat de bewerking binnen dezelfde centiseconde is gestart en gestopt. Maar wat gebeurt er als de start van de bewerking in de ene centiseconde valt en de bewerking in de volgende centiseconde wordt afgerond? In de volgende figuur zien we een aantal gangbare timingproblemen die zich voor kunnen doen als het meetresultaat in centiseconden worden uitgedrukt.



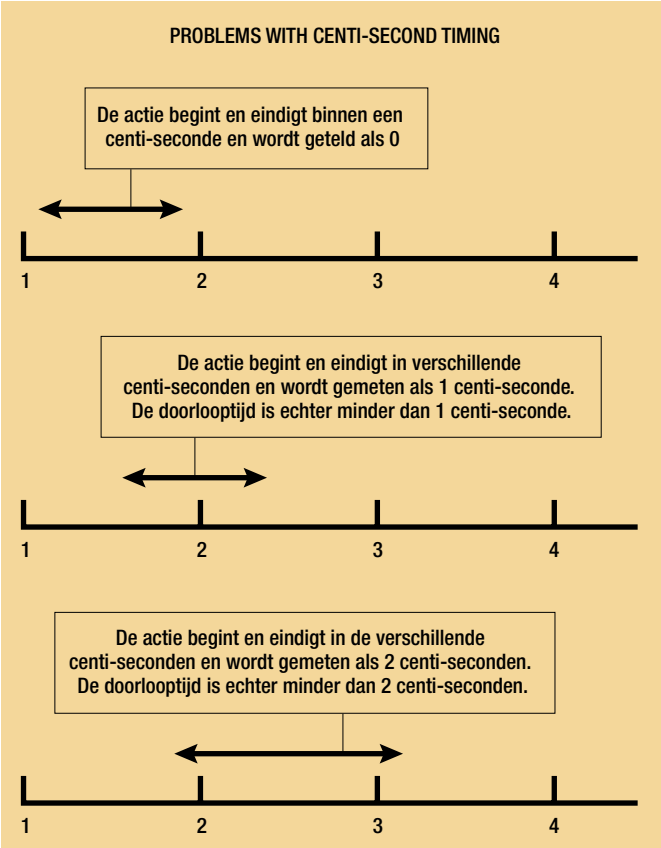
Figuur 1. Timingvraagstukken rond centiseconden

Het eerste blok laat een bewerking zien die binnen een interval van een centiseconde wordt getimed. Omdat begin en einde van de bewerking binnen dezelfde centiseconde vallen, krijgt de bewerking een doorlooptijd van 0 toegekend. Het tweede blok begint in een andere centiseconde dan waarin hij stopt; het blok kan korter zijn dan één centiseconde, maar wordt desondanks geteld als één centiseconde. Het laatste blok is verspreid over vier verschillende centiseconden, zodat de totale doorlooptijd drie centiseconden zou zijn, ook al is de werkelijke doorlooptijd net iets meer dan twee seconden. Wat dit alles ons laat zien, is dat bewerkingen soms te kort of te lang worden getimed.

Timing in Oracle vóór Oracle9i

Voordat Oracle9i Release 1 uitkwam, vond alle timing in Oracle plaats in centiseconden. Als gevolg van de voordurend toenemende snelheid van CPU's was deze meeteenheid niet meer echt bruikbaar, zoals we hierboven hebben uitgelegd. Een ander interessant punt is dat de kleinste time-out waarden in Oracle ook in centiseconden worden uitgedrukt. Dit kunnen we zien aan de hand van traces en informatie die we kunnen halen uit v$system_event en v$session_event[1]. Als er in Oracle een time-out optreedt, lijkt de minimumwaarde altijd één centiseconde te zijn. Als we aannemen dat de sessie zal worden

aangemeld, is dat niet echt een probleem. Als de sessie echter niet wordt aangemeld en een time-out moet krijgen alvorens door te kunnen gaan, kan er een hoop onnodig wachten ontstaan (zoals met de meeste latch waits het geval is).



Figuur 2. Problemen met timing in centi-seconden

Voordat Oracle9i verscheen was het ook onmogelijk om online de doorlooptijd en CPU-tijd van een SQL-statement te vinden. Het is bovendien onmogelijk te zien waarop een SQL-statement heeft staan wachten. In feite is de enige manier om dat te doen, een 10046 trace level 12[2] te activeren. Aan deze benadering kleeft een tweetal problemen:

- Een hoge overhead als gevolg van het naar process trace-bestanden schrijven van de trace-informatie;
- Kan, als er veel SQL moet worden uitgevoerd, een grote hoeveelheid disk/file system –opslag vereisen voor de process trace-bestanden.

[1] Bekijk, in releases van vóór Oracle9i de gemiddelde wachttijd in v$session_event en v$system_event, en in 'ela=' gegevens zoals die te vinden zijn in de door de '10046 level 12 trace' gegenereerde trace-bestanden.

[2] Alter session set events '10046 trace name context level 12, forever' fungeert om SQL-trace voor uw sessie te activeren en voor elke cursor alle waits en bind-informatie te laten zien.

[3] Alter system set timed_statistics=true.

Laten we eens kijken naar de in Oracle9i doorgevoerde veranderingen en de manier waarop ze performanceproblemen kunnen helpen oplossen.

Timing in Oracle9i Release 1

In Oracle9i Release 1 wordt de doorlooptijd uitgedrukt in microseconden. Dit maakt het een stuk eenvoudiger om nauwkeurig te laten zien waaraan Oracle zijn tijd besteedt. Time-outs worden nog altijd gedaan in centiseconden.. Een aantal voorgedefinieerde views is uitgebreid met mogelijkheden om in microseconden (en milliseconden) uitgedrukte tijdmetingen op te nemen en te verklaren. Alle timing binnen Oracle wordt bestuurd via de init.ora parameter *timed_statistics*. Deze parameter kan ook dynamisch op TRUE worden gezet, teneinde een Oracle-instance[3] te laten beginnen met het verzamelen van tijdsstatistieken.

Veranderingen in V$SQLAREA

V$SQLAREA is uitgebreid met twee nieuwe kolommen, namelijk CPU_TIME en ELAPSED_TIME, die beide een nauwkeurigheid van microseconden hebben. Hoewel de uitbreiding op zichzelf fraai is, brengt de volgende query twee interessante problemen aan het licht:

```
@c@Select cpu_time, elapsed_time
From v$sqlarea
Order by 2

  CPU_TIME  ELAPSED_TIME
-------------------------
    230000         174567
    260000         258803
    260000         271808
```

Allereerst is in twee van de drie rijen de doorlooptijd kleiner dan de CPU-tijd, en ten tweede wordt de CPU-tijd netjes afgerond. Het lijkt erop dat de CPU-tijd hier een afgeleide is van een timing in centiseconden.

V$SQLAREA

CPU_TIME	Number	Microseconden, maar afgeleid van centiseconden
ELAPSED TIME	Number	Op microseconden gebaseerd

De wachttijd voor het SQL-statement kan worden bepaald door de CPU_TIME af te trekken van de doorlooptijd ELAPSED_TIME. Het is onmogelijk vast te stellen waarop precies werd gewacht. De Oracle kernel beschikt in V$SYSTEM_EVENT over informatie over de instance-brede waits in Oracle (niet per SQL-statement,

maar niet over waits die zich in het besturingssysteem hebben voorgedaan (CPU waits, memory waits etc.).

Veranderingen in X$KSQST

Deze vaste tabel geeft informatie over de enqueues op je systeem. Vóór Oracle9i zouden dit alleen het aantal gets en aantal waits op enqueue-typen zijn geweest. Een enqueue bestaat uit twee characters en twee identifiers (id1 and id2). Een voorbeeld van een enqueue is TX, de transactie-enqueue. Met ingang van Oracle9i is het mogelijk om niet alleen de gets en de waits te zien, maar ook de wachttijd en het aantal keren dat een enqueue niet is gelukt (als gevolg van een interruptie of time-out).

X$KSQST		
KSQSTWTIM	Number	Dit getal lijkt op milliseconden te zijn gebaseerd

Er is echter nog altijd geen manier beschikbaar waarop we de enqueue terug kunnen voeren op individuele SQL-statements, en kunnen vaststellen hoe lang verschillende SQL-statements op een bepaalde enqueue hebben staan wachten.

Veranderingen in V$SYSSTAT

Deze vaste view bevat eigenlijk geen wijzigingen. Timing verloopt nog altijd in centiseconden, dus let op!

V$SYSSTAT	
CPU-gebruik door deze sessie	Centiseconden
Ontleedtijd cpu	Centiseconden
Doorlopen ontleedtijd	Centiseconden
Recursief cpu-gebruik	Centiseconden

Kijk uit bij het vergelijken van gegevens uit verschillende bronnen.

Veranderingen in V$WAITSTAT

Deze vaste view geeft een overzicht van de 'buffer busy' waits die zijn opgetreden, uitgesplitst naar blok-klasse. Er is een telling van het aantal waits en een tijd in microseconden.

V$WAITSTAT		
TIME	Number	Tijd in centiseconden

Het is niet mogelijk de waits in deze view tot een bepaald SQL-statement te herleiden.

Veranderingen in V$FILESTAT

Deze vaste view kent nu een splitsing tussen enkelvoudige block reads en het totale aantal reads. Dit maakt dus een betere splitsing op bestandsniveau mogelijk tussen enkelvoudige en meervoudige block reads.

V$FILESTAT		
MAXIORTM	Number	De maximale leestijd in centiseconden
MAXIOWTM	Number	De maximale schrijftijd in cent-seconden
SINGLEBLKRDS	Number	Het aantal enkelvoudige block reads
SINGLEBLKRDTIM	Number	De cumulatieve enkelvoudige block read time in centiseconden

Ook voor deze vaste view geldt dat het niet mogelijk is de reads tot een bepaald SQL-statement te herleiden.
Met behulp van de procedure dbms_system.kcfrms() is het mogelijk om de maximale lees- en schrijftijd te resetten, iets dat meer relevante intervalmetingen mogelijk maakt.

Veranderingen in V$SESSION_EVENT

De v$session_event legt de doorlooptijd van een wait nu vast in microseconden, en laat de waits vervolgens in microseconden en centiseconden zien. Door deze veranderingen is de wachttijd accurater (van de wachttijd af in microseconden en converteer dat getal naar centiseconden).

V$SESSION_EVENT		
TIME_WAITED_MICRO	Number	Doorlooptijd van de waits in microseconden
TIME_WAITED	Number	Afgeleid van time_waited_micro / 10000
MAX_WAIT	Number	Maximale wachttijd in centiseconden

Ook hier geldt echter dat het nog altijd niet mogelijk is om de waits te herleiden tot een specifiek SQL-statement in een bepaald tijdsinterval. Een andere interessante kolom die door DBA's meestal over het hoofd wordt gezien, is de MAX_WAIT kolom. Het is mogelijk te zien wat de maximale wachttijd voor een event is geweest. Deze informatie is geldig zolang een sessie is aangemeld en zal gedurende de levensduur van de sessie

de maximale wachttijd laten zien. Met behulp van de procedure dbms_system.kcfrms() kunnen we de maximumwaarden resetten en intervalmetingen voor deze kolom relevanter maken.

Veranderingen in V$SYSTEM_EVENT

De belangrijkste verandering is hier de toevoeging van de kolom TIME_WAITED_MICRO, die de doorlooptijd in microseconden laat zien. Als gevolg hiervan is de TIME_WAITED kolom nauwkeuriger geworden, aangezien deze van TIME_WAITED_MICRO is afgeleid.

V$SYSTEM_EVENT

TIME_WAITED_MICRO	Number	Doorlooptijd van de waits in microseconden
TIME_WAITED	Number	Centiseconden, afgeleid van time_waited_micro

SQL TRACE

Laten we eens kijken naar een trace die wordt gegenereerd door alter session set events '10046 trace name context level 12, forever' te doen:

```
@c@PARSING IN CURSOR #1 len=29 dep=0 uid=5 oct=3 lid=5
    tim=1006862415399006 hv=4384
51932 ad='84b05c04'
select count(*) from tab, tab
END OF STMT
PARSE #1:c=10000,e=11334,p=0,cr=4,cu=0,mis=1,r=0,dep=0,og=4,
    tim=1006862415398976
BINDS #1:
EXEC #1:c=0,e=587,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,
    tim=1006862415399931
WAIT #1: nam='SQL*Net message to client' ela= 10 p1=1650815232 p2=1 p3=0
FETCH #1:c=2500000,e=2526458,p=0,cr=149542,cu=0,mis=0,r=1,dep=0,og=4,
    tim=1006862
417926777
WAIT #1: nam='SQL*Net message from client' ela= 1004 p1=1650815232 p2=1 p3=0
FETCH #1:c=0,e=6,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=0,
    tim=1006862417928426
WAIT #1: nam='SQL*Net message to client' ela= 7 p1=1650815232 p2=1 p3=0
WAIT #1: nam='SQL*Net message from client' ela= 3636276 p1=1650815232 p2=1 p3=0
```

Merk op dat de tijdgerelateerde informatie in microseconden wordt gegeven. Ook hier zijn echter een paar interessante punten het bekijken waard:

1. Het veld *tim*= is uiteraard in microseconden. Het veld tim geeft een momentopname van de tijd.
2. De velden e= and *ela*= zijn uiteraard ook in microseconden. Beide velden geven een doorlooptijd weer.
3. Het interessante veld is ook hier *c*=. Ogenschijnlijk in microseconden, maar in feite weer afgeleid van een in centiseconden uitgedrukte waarde die met 10.000 is vermenigvuldigd. Het veld c geeft het CPU-gebruik weer.

Veranderingen in v$latch, v$latch_parent and v$latch_children

Voor Oracle9i lieten deze fixed views alleen informatie zien hoe vaak er op een latch gewacht werd, maar in Oracle9i laten ze nu ook zien hoe lang er gewacht wordt en wel in microseconden. De interessante kolommen in v$latch zijn:

Vóór Oracle9i was het mogelijk om de total_waits kolom in v$system_event voor het 'latch free' wait-event te koppelen aan de 'sleeps' in v$latch:

```
select event, total_waits, time_waited
from v$system_event
where event = 'latch free'
```

en

```
select name, trunc(sleeps*100/ total, 2) "Perc of Sleeps"
from v$latch, (select decode(sum(sleeps), 0, 1, sum(sleeps)) total
from v$latch)
where sleeps != 0
```

Een latch met veel sleeps zou dan evenredig veel wachttijd uit v$system_event kunnen hebben. In Oracle9i kunnen we dat preciezer zijn door rechtstreeks de WAIT_TIME te selecteren:

```
select name, trunc(wait_time*100/time_waited, 2) "Perc. of Time
Waited"
from v$latch, (select time_waited from v$system_event where event =
'latch free')
where wait_time != 0
order by 2
```

In Oracle9i zijn er 203 verschillende latches. Dit zijn zogenaamde 'parent-latches'. Een 'parent' kan 1 of meer 'children' hebben. De volgende query geeft wat meer informatie over de 'child latches':

NAME	COUNT(*)
SQL memory manager workarea list latch	67
Token Manager	2
buffer pool	8

cache buffers chains	1024
cache buffers lru chain	8
channel handle pool latch	1
channel operations parent latch	52
checkpoint queue latch	8
done queue latch	1
enqueue hash chains	1
global transaction	46
global tx hash mapping	47
granule operation	1
job workq parent latch	3
ksfv messages	2
latch wait list	1
library cache	1
message pool operations parent latch	11
msg queue latch	1
name-service namespace bucket	32
parallel query alloc buffer	4
parallel query stats	2
redo copy	2
resmgr:actses change group	150
resmgr:actses change state	150
resmgr:plan CPU method	2
resmgr:resource group CPU method	3
resmgr:session queuing	3
session idle bit	1
session queue latch	1
session switching	4
shared pool	7
simulator hash latch	32
simulator lru latch	8
temp table ageout allocation latch	1
tlcr meta context	1
tlcr state context list	1
virtual circuit buffers	170
virtual circuit queues	10

39 rows selected.

Dus negenendertig van de tweehonderddrie latches hebben een of meer child latches op het systeem waarop de query is uitgevoerd. Dit zal veranderen afhankelijk van een groot aantal factoren (zoals single versus multi cpu, init.ora parameters, opties in de tables, gebruikte features etc.). En de belangrijkste reden om 'child latches' te hebben, is om een betere concurrency te kunnen garanderen. Zo was er vóór Oracle9i bijvoorbeeld maar één shared pool latch beschikbaar. Nu zijn er meerdere (zie output). In ons voorbeeld is de shared pool verdeeld in zeven fysieke shared pools. Dit betekent dat voor veel installaties de contentie op de shared pool latch in Oracle9i wel eens verdwenen kan zijn. De fixed view v$latch_children kan dus meer informatie geven over welke child latch een probleem kan zijn. Vaak is het zo dat er grote contentie is op een van de child latches in plaats van een evenredige distributie van de contentie. V$latch children heeft dezelfde kolommen als v$latch. Dus in Oracle9i kun je nu ook de wachttijd zien voor een bepaalde child latch.

Wat is hetzelfde gebleven?

Alle andere timing informatie is in principe hetzelfde gebleven. Dus de informatie in V$SYSSTAT zoals "CPU used by this session", "parse time cpu", "recursive cpu usage", etc. zijn nog steeds in 1/100 sec. Wanneer Oracle deze waarden zou gaan weergeven in micro seconden, zouden vele reeds bestaande applicaties en tuningtools een probleem krijgen.

Conclusie

In Oracle9i is op een aantal strategische plaatsen uitgebreid met timing in microseconden, waarmee het mogelijk wordt om doorlooptijd en CPU-tijd met een nauwkeurigheid van microseconden te bekijken. De CPU-timing is in werkelijkheid echter niet in microseconden, maar wordt van centiseconden naar microseconden geconverteerd Dit is in elk geval vastgesteld bij Oracle9i voor Linux en SUN Solaris, en zou voor andere ports kunnen afwijken. De op de doorlooptijd gebaseerde kolommen (zoals wait_time) geven informatie in centiseconden, terwijl de statistieken in microseconden worden verzameld en vervolgens in centiseconden worden omgezet. Dit houdt in dat deze kolommen nauwkeuriger zijn De vaste views die Oracle biedt, maken het op dit moment nog niet mogelijk om specifieke wait-informatie voor een bepaald SQL-statement te bekijken.

Bronvermelding

Een deel van het materiaal waarvan voor dit artikel gebruik is gemaakt, is gebaseerd op e-mail-uitwisselingen tussen Cary Millsap (van Hotsos), Bjørn Engsing (van Miracle A/S), Mogens Nørgaard (van Miracle A/S) en Anjo Kolk.

Anjo Kolk
is chief Oracle technologist bij Precise Software Solution. Voordat hij bij Precise Software Solutions in dienst trad, werkte hij meer dan zestien jaar bij Oracle Corporation, waar hij zich vooral bezighield met rdbms-performance. Anjo is de schrijver van de bekende whitepaper 'Oracle7 wait events and enqueues', moderator van de website http://www.oraperf.com en geestelijk vader van YAPP methode (op responstijden gebaseerde fine tuning van Oracle systemen). Hij is per e-mail te bereiken op akolk@precise.com.