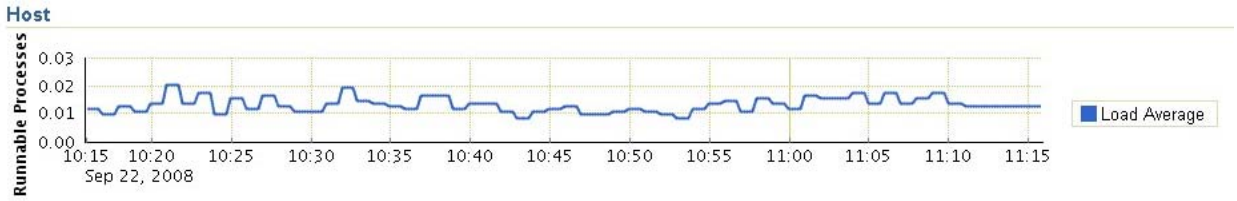
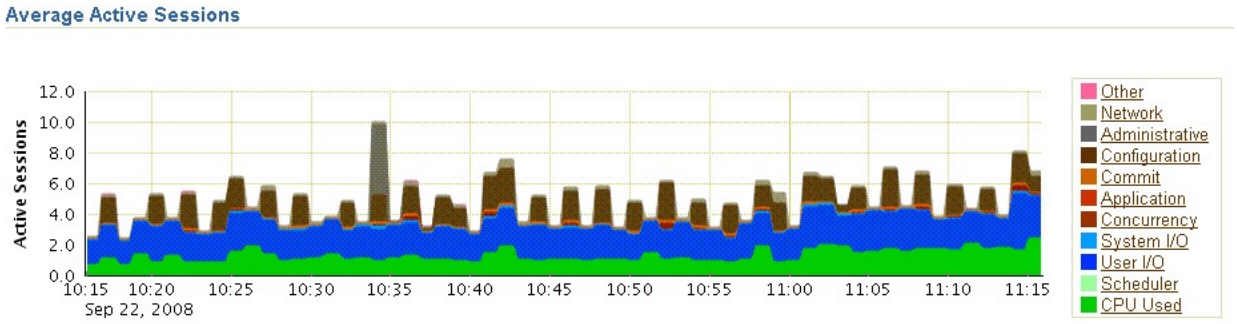
Real-time performance diagnose in Oracle

In Oracle 10g en 11g is het mogelijk om de database performance real-time te volgen, als u de licentie voor Diagnostics Pack heeft aangeschaft (en de Enterprise Edition database gebruikt).

Het Performance tabblad
In Enterprise Manager opent u een database als target en gaat dan naar de Performance tab. Wat u vervolgens ziet, geeft een algemene indruk van de load op het systeem, de database en I/O.



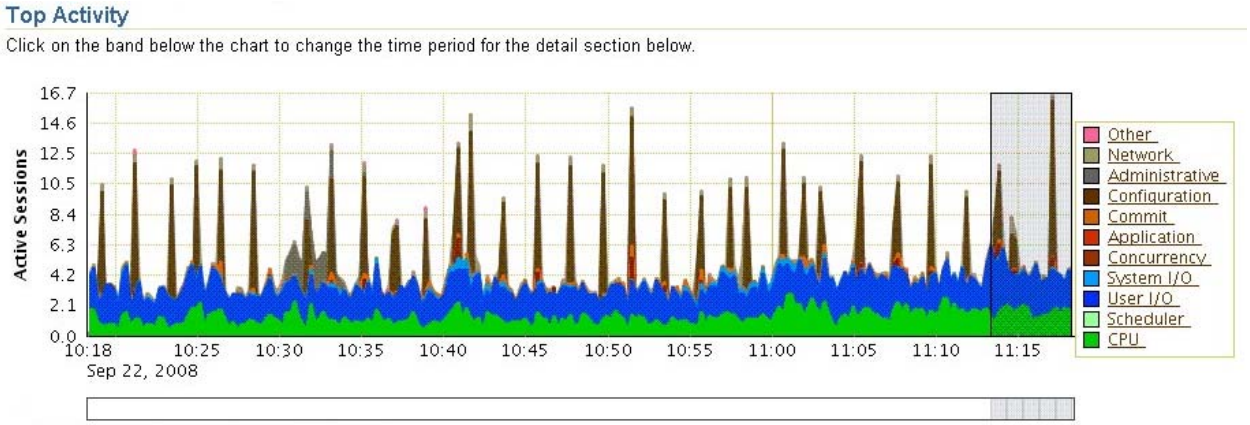
Figuur 1 De load op het systeem



Figuur 2 De verdeling van responstijd in de database

Extra details van de performance in de database kunt u zien met het Top Activity scherm dat u van hieruit kunt tonen.

Het Top Activity scherm
Deze grafiek wordt standaard elke 15 seconden ververst. Het laat in een oogopslag zien waar de responstijd van gebruikers in de database naartoe gaat. Oracle heeft dit onderverdeeld en toont het door middel van kleuren. Zo is CPU-gebruik groen, I/O is blauw, overmatig committen is oranje en problemen door de configuratie zijn bruin.



Figuur 3 Detail van de verdeling van de responstijd

In bovenstaande grafiek zien we bijvoorbeeld dat I/O de overhand heeft ten opzichte van het CPU gebruik. Dat is in normale gevallen niet echt wenselijk. Wat we ook zien, zijn pieken die bijna elke minuut voorkomen. De kleur is bruin, wat staat voor configuratie. Wanneer u moeite heeft de rode en bruine kleuren uit elkaar te houden, dan klikt u gewoon even in de legenda op een van de items. Ziet u dezelfde pieken (in andere kleuren overigens) bijvoorbeeld onder Configuration, dan is dat waar u verder moet zoeken.

Er zijn nog twee invalshoeken om het probleem te onderzoeken. In het Top Activity scherm kunt u namelijk ook op SQL statement of sessie zoeken. Dit deel van het scherm ziet er zo uit:

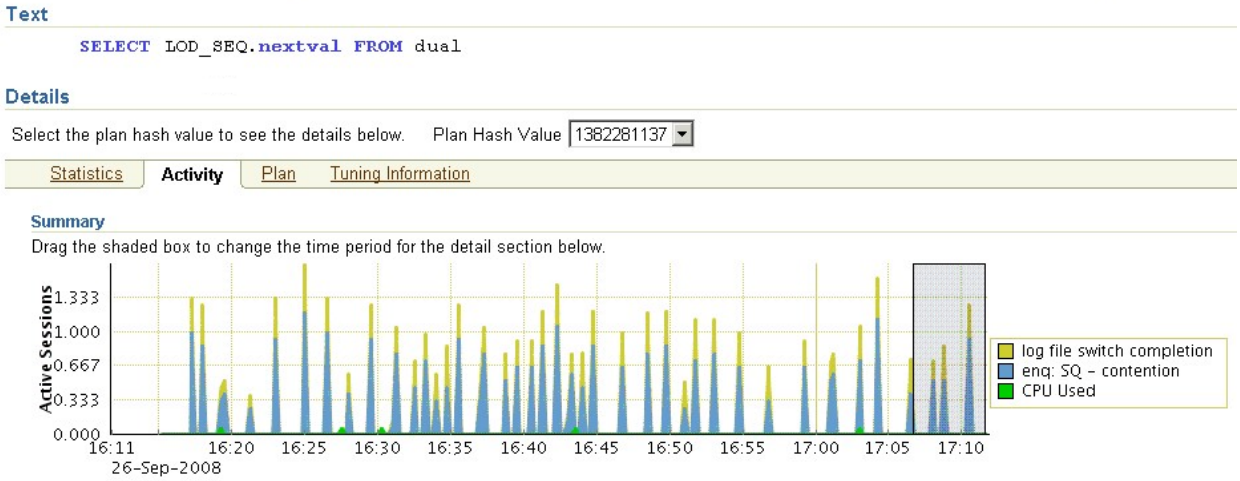
Figuur 4 De top SQL en sessies

Hier zien we de verdeling van responstijd per SQL en sessie. Wanneer u een zware SQL wilt onderzoeken, dan klikt u op het SQL ID. Wanneer u een sessie wilt onderzoeken, dan klikt u op het Session ID.

Wait events

Hoe u ook te werk gaat, u zult op een gegeven moment responstijdgegevens krijgen per zogenaamde “wait events”. Wait events geven aan waar de database mee bezig was, toen er gewacht werd. Oracle10g R1 bijvoorbeeld kent 806 verschillende wait events. Die hoeft u niet allemaal te kennen, maar het is handig van de belangrijkste te weten wat ze inhouden. Vandaar het overzicht aan het einde van dit artikel.

Met kennis van wait events kunt u de diagnose stellen van dit probleem. Een voorbeeld: het ophalen van een sequencenummer vertoonde dit responstijdprofiel:



Figuur 5 Responstijdprofiel per query

We zien twee wait events: log file switch completion en enq: SQL – contention, dit alles gecombineerd met CPU gebruik. De enq (enqueue, ofwel locking) is de grootste factor. Die onderzoeken we dus het eerst. In het overzicht aan het eind van het artikel zien we dat dit wait event met sequences te maken

heeft. Gelukkig kunnen we aan het SQL statement zien welke sequence het probleem geeft. Door zijn cache in te stellen of te vergroten, kunnen we ‘enq: SQ - contention' waits reduceren of vermijden.
De tweede wait event in dit voorbeeld, log file switch completion, was ook de belangrijkste oorzaak van de pieken in figuur 3, die in het zelfde systeem voorkwamen. Merk op in figuur 3 hoe deze pieken bijna elke 2-4 minuten voorkomen. Oracle wacht op log file switch completion als er een redo log switch plaats vindt. Als dat op een OLTP productiesysteem elke 2-4 minuten voorkomt, dan is dat vrij fors. Door de redo logfiles te vergroten, kunnen we deze wait event reduceren.

Overzicht wait events

Naam wait event	**Achtergrond**
buffer busy wait	Twee processen trachten op bijna hetzelfde moment hetzelfde block te krijgen en dat block zit niet in de buffer cache. Het ene proces zal het block in de buffer cache lezen en blokkeren. Het andere moet wachten en daardoor treedt dit wait event op. Aanpak: vergroot de db buffer cache.
db file scattered read	Dit treedt op bij multiblock reads. Multiblock reads treden op bij full table scans en index fast full scans. Als het SQL ook daarwerkelijk deze data nodig heeft hoeft dat niet slecht te zijn. Aanpak: Deze wachttijd kan verminderd worden door óf de I/O te versnellen (andere indeling disks, direct I/O, etc.) óf (als multiblock reads op deze blokken vaker voor komen) de buffer cache te vergroten. Afhankelijk van de applicatie kan ook de parameter db_multiblock_read_count vergroot worden. Dit is in het voordeel voor batchprocessen, rapportage databases en datawarehouses. Wees wel voorzichtig met deze parameter: hij heeft rechtstreeks gevolg voor het gedrag van de optimizer en Oracle zet hem vanaf versie 10g automatisch.
db file sequential read	Een Oracle proces wil een blok lezen dat niet in de buffer cache zit, dus moet het van disk gelezen worden. Aanpak: Indien het gaat om willekeurige leesacties, dan heeft het geen zin om de buffer cache te vergroten. Dan moet men zien de I/O te versnellen (andere indeling disks, direct I/O, etc.). Meestal treedt dit event op als bijverschijnsel van full table scans of index fast full scans, die weer een teken kunnen zijn voor minder optimale SQL.
direct path read	Deze waits worden veroorzaakt door SQL statements die direct read operaties uitvoeren op tablespaces. Vaak zijn dit sorteeracties (ORDER BY, GROUP BY, UNION, DISTINCT en ROLLUP) die plaats vinden in de temporary tablespace. Oracle probeert sorteeracties eerst in de work area van de PGA uit te voeren. Als daar niet genoeg ruimte is, dan zal de temporary tablespace worden aangesproken. Ook als Oracle hash joins uitvoert en de hash partities passen niet in de work area van de PGA, dan wordt de temporary tablespace gebruikt. Een laatste mogelijkheid is dat de sessie waar direct path read optreedt zich bedient van parallel execution. Aanpak: Kijk of het betrokken bestand een temporary tablespace is (de eerste parameter die bij dit wait event hoort, is het file_id). Indien de parameter workarea_size_policy op AUTO staat (wordt aangeraden), verhoog dan de parameter pga_aggregate_target.
enqueue	Bij enqueue is er sprake van locking. Meerdere sessies proberen een

	gedeelde bron tegelijkertijd te gebruiken. Aanpak: Aan de hand van de parameters die bij deze wait event horen is af te leiden wat voor lock het betreft. Of gebruik views als v$locked_object, v$lock of v$enqueue_stat. In Oracle 10g en 11g is enqueue verdeeld in verschillende enq wait events met beschrijvende namen.
enq: SQ - contention	Er wordt gewacht op een nieuw nummer van een sequence en Oracle kan die niet snel genoeg geven. Aanpak: Oracle kan sequence nummers cachen in het geheugen. Met ALTER SEQUENCE <naam> CACHE 20 kunt u 20 sequence nummers cachen. Wanneer de instance uitvalt als er sequencenummers in het geheugen staan, dan is Oracle deze sequencenummers kwijt. Het kan dus zijn dat er na opstarten een “gat” is in de gebruikte ID’s bijvoorbeeld. Dat zou geen probleem moeten zijn voor een applicatie, maar een enkele kan er niet tegen.
free buffer wait	Voordat Oracle een block kan lezen in de buffer cache, kijkt het of er een block vrij is. Indien dat er niet is, dan treedt dit wait event op. Dat kan twee oorzaken hebben: of de buffer cache is niet groot genoeg. Of de DBWR schrijft niet snel genoeg blocks weg, waardoor er in de LRU lijst niet snel genoeg nieuwe lege buffers komen. Aanpak: onderzoek de performance van de I/O. Vergroot eventueel de buffer cache.
log buffer space	Er zijn twee mogelijke oorzaken van dit event: of de log buffer is te klein, of de LGWR is te traag. Over het algemeen is de default log_buffer parameter (512 Kb) overigens prima. Als log_buffer te groot is, kan het namelijk weer log file sync waits veroorzaken. Aanpak: Kijk of er operaties op objecten zijn waarover geen redo gegenereerd hoeft te worden, bijv. op staging tabellen in datawarehouses. Bedenk dat NOLOGGING redo generatie alleen overslaat bij direct load insert operaties. Deletes en updates worden wel meegenomen en dat is een risico bij een eventuele restore. Daarbij heeft het gebruik van NOLOGGING mogelijke consequenties voor het gebruik van standby databases.
latch free	Latches zijn een soort locks op data structuren in de SGA. Ze voorkomen dat meerdere sessies tegelijkertijd hetzelfde deel van het SGA wijzigen. Het latch free wait event kon in Oracle 9i en daarvoor nog van alles betekenen. In Oracle 10g en 11g is het latch free wait event onderverdeeld in een aantal beter beschrijvende namen. Aanpak: onderzoek welke latch het betreft en zoek op of probeer te beredeneren waarom de applicatie niet bij de latch kan.
log file sync	Met log file sync wordt bedoeld: de synchronisatie van de log buffer met de log file. Deze wait event komt dus voor als Oracle die niet kan bijhouden. Dat gebeurt onder andere als een applicatie te vaak commit. Ook een veel te grote log_buffer parameter kan zorgen de sessies in de database steeds een tijd moeten wachten tot de log file is bijgewerkt. En niet te vergeten kan te trage I/O bij het wegschrijven naar de redo log file voor problemen zorgen.

	Aanpak: Onderzoek of er te veel commits plaatsvinden of dat de log_buffer te groot is ten opzichte van de redo logfiles.
log file switch completion	Er wordt zoveel redo gegenereerd, dat redo log files regelmatig switchen. Dat is een relatief duur proces waar sessies op moeten wachten, dus het is beter dat niet te vaak te moeten doen. Aanpak: Door de redo log te vergroten hoeven switches niet zo vaak voor te komen. Deze oplossing heeft wel gevolgen voor de recovery strategie. Want het duurt langer voor er een archivelog weggeschreven wordt, zodat by een crash relatief veel transacties in de online redo log verloren kunnen gaan.
SQL*Net message from client SQL*Net message to client	Hoewel veelal beschouwd als “idle” event, kunnen deze wait events wel degelijk wat te betekenen hebben. SQL*Net message from/to client beslaat in feite alle wait tijd vanaf het moment dat communicatie naar de client gaat tot het moment dat een nieuwe opdracht terugkomt. Applicaties die hier grote waits hebben, kunnen bijvoorbeeld zelf verder werk doen met de data. Veelal kan men daar beter voor terecht bij Oracle. Een andere mogelijke oorzaak is netwerk latency. De oorzaak zou zelfs kunnen zitten in SQL*Net zelf of zijn pakketgrootte. Dit kan bijvoorbeeld gebeuren als een batchproces gestart wordt vanaf de database server en connect aan de database op diezelfde server. Aanpak: Omdat batches applicatief zijn, is het aan te raden om ze op een database server te draaien. Connect liever met het bequeath protocol en niet via SQL*Net.