ALMG / GTI / GTEC

SGBD Oracle – Coleta e análise de estatísticas

Ferramenta Open Source para Análise de Desempenho do Banco de Dados Oracle

Gerência de Tecnologia e Rede (GTec)
Gerência-Geral de Tecnologia da Informação (GTI)
Assembleia Legislativa de Minas Gerais (ALMG)

ALMG / GTI / GTEC

SGBD Oracle – Coleta e análise de estatísticas

Sumário

ALMG / GTI / GTEC

SGBD Oracle – Coleta e análise de estatísticas

1 Estatísticas de desempenho do Oracle

A licença da versão Standard Edition (SE), Standard Edition 1 e 2 (SE1 / SE2) do Oracle não permite que a principal ferramenta de monitoração de desempenho do banco de dados seja utilizada: o Automatic Workload Repository (AWR). Esta ferramenta persiste dados de várias views de desempenho do banco em intervalos de tempo (*snapshots*), para que depois esses dados possam ser tratados e analisados [1].

Desde a versão 8i [2], a Oracle disponibiliza o Statspack, que não requer licença adicional. Desta forma, usuários da versão Standard podem utilizar esta ferramenta para monitoração de desempenho do banco. De maneira similar ao AWR, o Statspack coleta informações de algumas views de desempenho do Oracle. As diferenças são decorrentes da disponibilidade do Active Session History, além de outras features que são específicas do AWR[3].

Em alguns casos, os dados do statspack podem apresentar mais informações do que se deseja. A execução dos snapshots possui um alto custo, consumindo uma quantidade considerável de recursos do banco. O Oracle já define um número predefinido de estatísticas que são coletadas a partir do parâmetro i_snap_level [4].

A demanda local era de **verificar e acompanhar o comportamento dos programas e módulos que utilizam o Oracle, ao longo do dia, em relação a três métricas principais**: CPU, Leituras e Escritas. Além disso, houve a necessidade de medir o consumo de diferentes aplicações web, que utilizam o *JDBC Thin Client* em relação a essas três métricas.

Para isso, os comportamentos de sessões específicas foram analisados, utilizando as views de desempenho e recursos como scripts, *triggers* e *procedures* para registrar *snapshots* dos dados dessas views, sem que houvesse utilização de objetos não permitidos pela licença do Oracle Standard.

Além das estatísticas de sessões, é essencial que estatísticas globais do banco sejam coletadas, de forma a registrar o comportamento não apenas das aplicações, mas também do próprio banco como um todo.

2 Obtenção das estatísticas de sessões

2.1 Sessões do Oracle (V$SESSION / GV$SESSION)

O Oracle pode ser utilizado por vários tipos diferentes de aplicações, como executáveis Delphi, planilhas de excel ou por um servidor de aplicação como o Tomcat, Wildfly, etc. Cada um dos tipos de aplicação possui formas específicas de conexão. As sessões ativas podem ser consultadas na view **V$SESSION** (ou **GV$SESSION** no Oracle RAC).

Aplicações são identificadas pelo atributo PROGRAM e pelo atributo MODULE da V$SESSION (**nem sempre os atributos possuem o mesmo valor**). No caso de aplicações executáveis (binários), o nome é mantido durante toda a sessão. No entanto, existem casos em que o nome do módulo da aplicação pode ser alterado, através do DBMS_INFO [5], fazendo com que uma única sessão tenha mais de um nome de módulo/aplicação durante seu tempo de conexão).

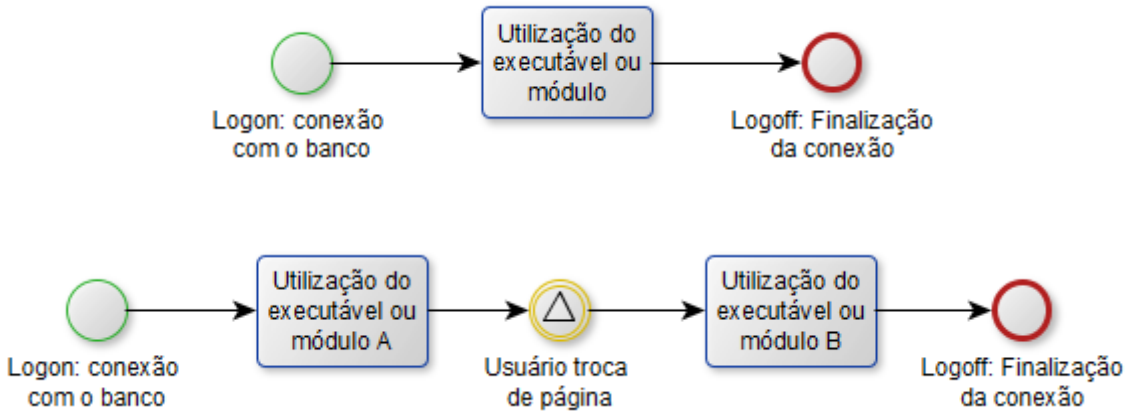


Figura 1: Sequência de eventos de conexão do Oracle

Todas as sessões possuem o evento de LOGON, momento em que a conexão do usuário com o banco é estabelecida e LOGOFF, momento em que o usuário libera o recurso de conexão para outra.

Cada sessão possui um único identificador em uma instância (lembrando que o Oracle RAC pode possuir mais de uma instância ativa). Logo, uma conexão pode ser identificada pela tupla formada pelos atributos (INST_ID, SID) da tabela GV$SESSION.

Após o término de uma conexão, o SID é liberado e fica disponível para ser utilizado por outra conexão. Logo, **pode haver vários registros distintos de uma tupla (INST_ID, SID)** ao longo da execução do servidor Oracle. Atributos que podem diferenciar a conexão é o atributo LOGON_TIME e os atributos PROGRAM e MODULE.

2.2 Snapshot das sessões (TESTATISTICAS_RAC)

Além das views de sessão V$SESSION / GV$SESSION, o Oracle disponibiliza views de estatísticas de cada uma das sessões: a GV/V$SESSTAT. Essas views possuem informações de utilização de recursos de banco por cada uma das sessões ativas no banco (que estejam na GV/V$SESSION. A view V$MYSTAT possui informações da sessão específica que está ativa.

Todas essas views possuem contadores incrementais. GV/V$SESSTAT e V$MYSTAT registram o consumo total de uma sessão desde a hora que ela foi criada (conexão estabelecida). Logo, é possível fazer *snapshots* dessas views. Para identificar o nome da aplicação ativa no

momento, dados da GV/V$SESSION também devem ser registrados. A tabela contendo o registro das sessões é a **TESTATISTICAS_RAC** e possui a seguinte estrutura:

```
TESTATISTICAS_RAC(
  INST_ID     NUMBER         NOT NULL,    -- GV$SESSION
  SID         NUMBER         NOT NULL,    -- GV$SESSION
  SCHEMANAME  VARCHAR2(32)   NOT NULL,    -- GV$SESSION
  TIMESTAMP   DATE           NOT NULL,    -- SYSDATE
  STATISTIC#  NUMBER         NOT NULL,    -- GV$SESSTAT
  VALUE       NUMBER         NOT NULL,    -- GV$SESSTAT
  MACHINE     VARCHAR2(64),               -- GV$SESSION
  PROGRAM     VARCHAR2(64),               -- GV$SESSION/TRATADO
  EVENT       NUMBER         NOT NULL,    -- <EVENTO>
  LOGON_TIME  DATE                        -- GV$SESSION
);
```

- Atributos da view GV$SESSION:
  - INST_ID: instância do Oracle;
  - SID: número de identificação da sessão na instância;
  - SCHEMANAME: usuário do banco utilizando a sessão;
  - MACHINE: hostname conectado ao Oracle;
  - LOGON_TIME: horário de logon da sessão.
- Atributos da view GV$SESSTAT:
  - STATISTIC#: identificador numérico da estatística, conforme [6] e [7].
  - VALUE: valor acumulado da estatística.
- Atributos tratados ou adicionados:
  - PROGRAM é um condicional dos atributos PROGRAM, MODULE e MACHINE da GV$SESSION.
  - TIMESTAMP é o momento de coleta da estatística (SYSDATE no momento da coleta).
  - EVENT é a definição do evento de coleta de estatística.

Os atributos tratados e/ou adicionados foram incluídos na tabela com os seguintes propósitos:

- TIMESTAMP: indica o momento em que a estatística foi coletada. Uma conexão pode durar um segundo ou várias horas (caso for persistente). A utilização do atributo TIMESTAMP funciona como se fosse um “snapshot id” para uma determinada coleta.

- PROGRAM: identificar o nome do programa ou módulo consumindo os recursos do banco. O valor depende das aplicações que se conectam ao banco, podendo ser obtido diretamente de GV/V$SESSION ou sendo tratada. **Deve-se lembrar que a execução de DBMS_APPLICATION_INFO.SET_MODULE altera o atributo MODULE de GV/V$SESSION e não o atributo PROGRAM**. Um exemplo de processamento desse valor, verificando atributos PROGRAM, MODULE e MACHINE de GV$SESSION seria:

```
INSERT INTO TESTATISTICAS_RAC (... PROGRAM, ...)
SELECT INST_ID, SID, ... ...
  CASE WHEN PROGRAM = 'JDBC Thin Client' THEN
    CASE
      WHEN MACHINE = 'host_intranet' THEN 'JDBC Intranet'
      WHEN MACHINE = 'host_internet' THEN 'JDBC Internet'
      ELSE 'JDBC from '||MACHINE
    END
  ELSE
    CASE WHEN PROGRAM IS NULL THEN
      CASE
        WHEN MODULE IS NULL THEN 'NULL'
        ELSE MODULE
      END
    ELSE PROGRAM
    END
  END as PROGRAM,
  ...
FROM
  GV$SESSION A,
  GV$SESSTAT B
WHERE ...
```

O exemplo acima verifica se o programa é um conector Java do Oracle. Se for, o programa pode ser um módulo da Intranet (portal interno), da Internet (portal público) ou de algum servidor de desenvolvimento ou cliente desktop. Se não é um conector Java, ele define como sendo o próprio valor de PROGRAM, se não for nulo ou de módulo, este não for nulo ou a string NULO se ambos PROGRAM e MODULE tiverem valor NULL.

- EVENT: identificar em qual momento a coleta de estatística é executada. O número do evento é importante para análise e processamento do consumo de recursos através dos registros gerados.

EVENT	Método de registro	Quando
0	trigger trg_logon	LOGON do usuário
1	script roda-amostragem.sh + roda-amostragem.sql	A cada 10 minutos
2	procedure prosetanomemodulo	Registro do módulo a ser substituído. Via de regra, acontece na intranet.
3	procedure prosetanomemodulo	Registro do módulo chamado. Via de regra, acontece na intranet.
4	trigger trg_logoff	Trigger de logoff é executada.

Tabela 1: Eventos de coleta de estatísticas do Oracle

Os códigos e descrição dos eventos são gravados na tabela TESTATISTICAS_EVENTOS. As estatísticas inseridas na tabela TESTATISTICAS_RAC e suas descrições ficam registradas TTIPO_ESTATISTICA_RAC. **Para incluir alguma estatística**, é necessário que o número identificado por STATISTIC# tenha sua descrição na tabela V$STATNAME [6]. Por padrão, escolheu-se as métricas "session logical reads", "CPU used by this session" e "db block changes".

2.2.1 Método de coleta das estatísticas por evento

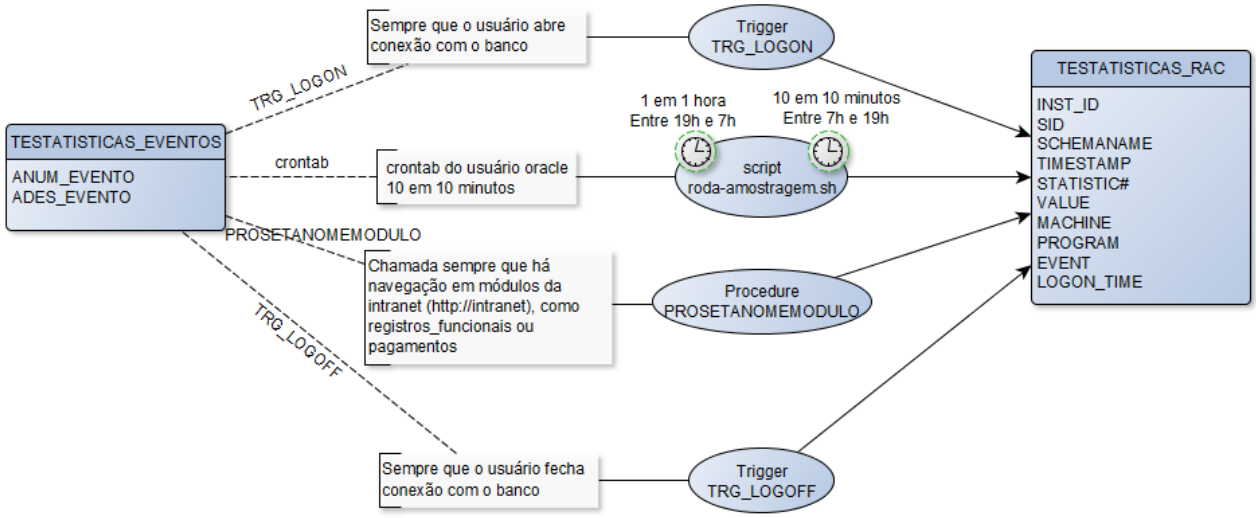
- **Evento 0**: LOGON. O Oracle possui uma trigger que pode ser executada no instante em que um usuário se autentica no banco e inicia uma sessão [8]. A trigger **TRG_LOGON** executa a inserção dos dados em TESTATISTICAS_RAC.

- **Evento 1**: execução periódica e agendada de script. De 10 em 10 minutos (ou no período que o DBA desejar), os scripts **roda-amostragem.sh** e roda-amostragem.sql fazem um snapshot das views de desempenho para TESTATISTICAS_RAC.

- **Eventos 2 e 3**: executados em ambiente web. Ex: na intranet, há os módulos RH e FIN. Quando um usuário com acesso aos dois módulos sai da página de RH e vai para a página FIN, um procedimento é executado para registrar a interrupção do consumo do módulo RH e início do módulo FIN. Essa transição é feita pela procedure **PROSETANOMEMODULO** e **deve ser incorporada pelos desenvolvedores web** que utilizam o conector java JDBC Thin Client.

- **Evento 4**: LOGOFF. Similar ao evento 0, mas para o caso de LOGOFF do usuário. A trigger referente é a **TRG_LOGOFF**.

Desta forma, a tabela TESTATISTICAS RAC é alimentada pelos seguintes processos:



2.3 Análise das estatísticas

Os registros da tabela TESTATISTICAS_RAC são *snapshots* das views de desempenho ao longo do tempo. Uma vez que essas views possuem contadores acumulados, é necessário tratar esses dados um a um, de forma a obter os valores corretos para a análise que se deseja fazer.

Todas as análises envolvem o nome do programa. Os dois tipos principais de relatório são:

1. Relatório de **utilização de programas por hora**: identifica os maiores consumidores de recursos em um determinado período de tempo, medido em granularidade de 1h.

2. Relatório de **utilização de programas por sessão**: identifica as sessões que mais consumiram recursos. No caso de sessões, não faz sentido a utilização do atributo "timestamp", pois uma sessão pode durar poucos segundos ou várias horas.

- Observação: é possível adicionar / alterar métodos da análise, alterando o código-fonte da aplicação.

Os relatórios gerados podem incluir ou excluir a análise dos registros compreendidos entre 22:15 e 07:30, uma vez que pode haver casos de backup, indisponibilidade da instância ou simplesmente o consumo do banco fora deste horário não ser importante. Por padrão, os seguintes agendamentos (via crontab) são feitos:

Período	Scripts	Tabela Gerada	Noturno?	Hist?
5 minutos	analisa_sessoes.sh	TANALISE_ESTATISTICAS_SESSOES	Se h < 8	Não
10 minutos	sumariza-amostragem-perl.sh CURRENT	TANALISE_ESTATISTICAS	Sim	Não
22:30	sumariza-amostragem-perl.sh WORKTIME	TANALISE_ESTATISTICAS_HORA	Não	Sim
00:05	sumariza-amostragem-perl.sh FULLDAY	TANALISE_ESTATISTICAS_HORA24	Sim	Sim

Tabela 2: Execução dos scripts de análise / sumarização

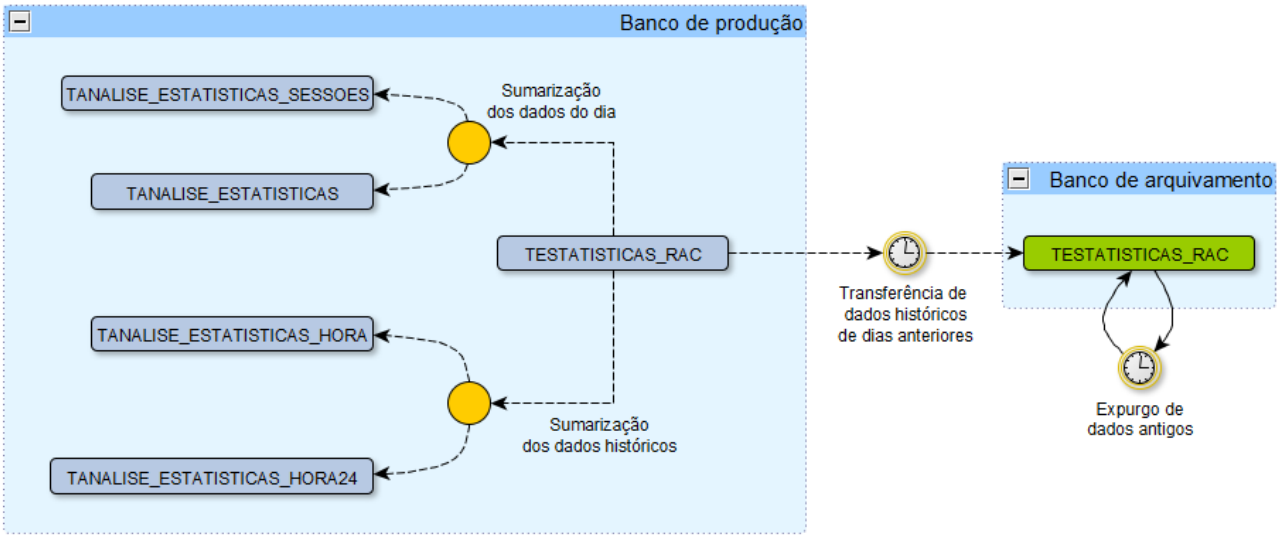
- Noturno: dados da TESTATISTICAS_RAC com processamento caso TIMESTAMP > 22:15 e TIMESTAMP < 07:30;
- Hist: possui dados históricos (de dias anteriores).

2.4 Sumarização e transferência da tabela de estatísticas

O número de registros gerados na tabela TESTATISTICAS_RAC é muito grande, podendo ser mais de 1M por dia, dependendo da atividade do banco e da quantidade de estatísticas geradas. Para

evitar que o banco seja onerado com as análises e que estas demorem muito para ser executadas, é essencial que dados mais antigos sejam sumarizados e/ou transferidos para outro banco de dados.

Uma possível solução é executar um processo que sumariza os dados do dia anterior em outras tabelas (conforme Tabela 2) e migrar os dados dos dias anteriores para outro banco de dados.



Os dados antigos podem ser transferidos para outro banco de dados (como outra instância do Oracle, postgresql ou mysql) ou para arquivos (csv, csv comprimido, dump do Oracle, etc). Como o volume é muito grande, é importante que haja um processo de expurgo de dados mais antigos (ex: dados gerados há mais de um ano).

3 Processo de análise estatística

3.1 Tipos de consulta

Conforme descrito nos itens de Análise das estatísticas, os relatórios de utilização dos programas podem ser divididos em utilização por sessão e utilização por hora. Segue um exemplo (simplificado em relação à estrutura de TESTATISITCAS_RAC) de cada uma das análises:

INST_ID	SID/TL	PROGRAM	TIMESTAMP	STATISTIC#	VALUE	EVENT
1	22/09:00	P1.EXE	2018-09-14 09:00	19	0	0
1	22/09:00	P1.EXE	2018-09-14 09:00	19	10	1
1	22/09:00	P1.EXE	2018-09-14 09:20	19	17	1
1	22/09:00	P1.EXE	2018-09-14 09:40	19	35	4
1	50/09:45	P1.EXE	2018-09-14 09:45	19	0	0

1	50/09:45	P1.EXE	2018-09-14 09:55	19	10	1
1	50/09:45	P1.EXE	2018-09-14 10:05	19	23	1
1	50/09:45	P1.EXE	2018-09-14 10:15	19	38	1
1	50/09:45	P1.EXE	2018-09-14 10:25	19	40	4

Tabela 3: Exemplo de coleta de dados de estatísticas. TL (Time Logon) foi incluído junto ao SID, pois é um dos atributos que identificam uma determinada sessão.

A análise do programa P1.EXE **por sessão** é delimitada pelo início e fim de cada uma das sessões. Supondo que seja um cursor ou que um programa faça o loop em todas as linhas da tabela, um indicativo de que uma nova sessão é iniciada é quando o último evento foi um logoff (evento 4). A execução do analisador por sessão seria:

INST_ID, SID	TIMESTAMP	INCREMENTO	VALOR[1, 22, 09:00]	VALOR[1, 50, 09:45]
1, 22, 09:00	N/A	0	0	0
1, 22, 09:00	N/A	10	10	0
1, 22, 09:00	N/A	7	17	0
1, 22, 09:00	N/A	18	35	0
1, 50, 09:45	N/A	0	35	0
1, 50, 09:45	N/A	10	35	10
1, 50, 09:45	N/A	13	35	23
1, 50, 09:45	N/A	15	35	38
1, 50, 09:45	N/A	2	35	40

VALOR[P1.EXE, 1, 22, 09:00] = 35
VALOR[P1.EXE, 1, 50, 09:45] = 40

Já a análise de P1.EXE **por hora** pode ser feito da seguinte forma:

INST_ID	SID/TL	TIMESTAMP	VALUE	INCREMENTO	EVENT
1	22/09:00	**2018-09-14 09**:00	0	0	0
1	22/09:00	**2018-09-14 09**:00	10	10	1
1	22/09:00	**2018-09-14 09**:20	17	7	1
1	22/09:00	**2018-09-14 09**:40	35	18	4
1	50/09:45	**2018-09-14 09**:45	0	0	0
1	50/09:45	**2018-09-14 09**:55	10	10	1
1	50/09:45	**2018-09-14 10**:05	23	13	1

1	50/09:45	**2018-09-14 10**:15	38	15	1
1	50/09:45	**2018-09-14 10**:25	40	2	4

VALOR[P1.EXE, 2018-09-14-09] = 45
VALOR[P1.EXE, 2018-09-14-10] = 30

Os exemplos acima são muito simplificados. A tabela nem sempre possui os registros de uma mesma sessão de forma adjacente, os SIDs podem ser reaproveitados e um módulo pode ter o nome alterado em um mesmo SID. Logo, é necessário tratar todos esses casos utilizando ordenação na consulta e tratamento/processamento das linhas retornadas.

3.2 Consulta em janela na tabela TESTATISTICAS_RAC

A consulta principal para ambos os tipos de sumarização / relatório é a seguinte (as linhas em azul devem ser parametrizadas no programa que executa a análise das estatísticas):

```
SELECT
  rowid, inst_id, sid, program, schemaname, machine, timestamp,
  extract(year from timestamp) as year,
  extract(month from timestamp) as month,
  extract(day from timestamp) as day,
  extract(hour from cast(timestamp as timestamp)) as hour,
  extract(minute from cast(timestamp as timestamp)) as minute,
  statistic#,
  value,
  coalesce(lag(value) over (
    partition by statistic#, inst_id, sid, schemaname
    order by timestamp, value, event),
    0) as lastval,
  coalesce(value - lag(value) over (
    partition by statistic#, inst_id, sid, schemaname
    order by timestamp, value, event),
    0) as delta, event,
  logon_time
FROM TESTATISTICAS_RAC
WHERE
  timestamp >= TO_DATE('2018-08-29 00:00', 'YYYY-MM-DD HH24:MI')
  AND timestamp <  TO_DATE('2018-08-30 00:00', 'YYYY-MM-DD HH24:MI')
  AND statistic# = 19
  AND UPPER(program) IN ('PROG1.EXE', 'PROGWEB')
ORDER BY
  statistic#, inst_id, sid, schemaname, logon_time, timestamp, value, event
```

Texto 1: Consulta SQL para obtenção dos valores de TESTATISTICAS_RAC

Na consulta, o atributo STATISTIC# é utilizado como primeiro argumento de ordenação para separar as diferentes estatísticas que estão sendo coletadas (utilização de CPU, leituras, escritas, etc). Os atributos INST_ID, SID, SCHEMANAME e LOGON_TIME são usados logo a seguir, para separar cada uma das sessões (sendo possível verificar o início e término de cada uma). O restante da ordenação (TIMESTAMP, VALUE, EVENT) é referente aos registros internos de uma mesma sessão.

A função de janela LAG (value) OVER (PARTITION BY statistic#, inst_id, sid, schemaname) é utilizada para auxiliar no processo de análise. O custo da utilização da função de janela não onera a consulta, que já foi ordenada. O custo da query com e sem a função LAG foi verificado com a execução utilizando a HINT do Oracle /*+ gather_plan_statistics */ e a consulta do plano de execução:

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(
 '<SQL_ID>',  NULL, FORMAT =>'ALLSTATS LAST +cost +bytes'))
```

onde SQL_ID é obtido pesquisando a consulta em GV/V$SQL.

3.3 Processamento do resultado da consulta

O processamento do resultado devolvido pela consulta depende de qual tipo de consulta foi feita: por sessão ou por hora (conforme exemplos em 3.1 Tipos de consulta). Para cada linha processada, deve-se escolher qual é o método de agregação e qual deve ser o incremento para a linha agregada.

- Análise por sessão: agregar INST_ID, SID, UPPER(SCHEMANAME), UPPER(PROGRAM), MACHINE, STATISTIC#, TIMESTAMP(YYYY-MM-DD), LOGON_TIME.

- Análise por hora: agregar UPPER(PROGRAM), STATISTIC#, TIMESTAMP(YYYY-MM-DD HH)

A agregação pode ser uma chave de hash, cujo valor é incrementado sempre que a sequência de atributos identificar a chave. Exemplo:

- Processamento por sessão:
  chave = [INST_ID, SID, UPPER(SCHEMANAME), UPPER(PROGRAM), MACHINE, STATISTIC#, TIMESTAMP(YYYY-MM-DD), LOGON_TIME]
  valor[chave] = valor[chave] + incremento

- Processamento por hora:
  chave = [UPPER(PROGRAM), STATISTIC#, TIMESTAMP(YYYY-MM-DD HH)]
  valor[chave] = valor[chave] + incremento.

O valor de “incremento” depende de vários fatores para ser calculado. Como a tabela TESTATISTICAS_RAC possui várias fontes distintas, não basta que o valor do incremento seja o

valor do atributo delta, calculado na consulta como

```
coalesce(value - lag(value) over (
      partition by statistic#, inst_id, sid, schemaname
      order by timestamp, value, event),
      0) as delta
```

As condições que interferem na decisão do valor a ser incrementado para o elemento (chave) atualmente processado são apresentadas abaixo. As variáveis são os valores obtidos na consulta Texto 1.

1. Se **sid ≠ last_sid**: houve a interrupção de uma sessão.
   1. Se **event == LOGOFF, incremento = valor**. Ocorre quando a sessão possui apenas um registro. Mantido por compatibilidade dos registros até 03/08/2018, pois nas primeiras versões do programa não existia a trigger de logon, apenas de logoff.
   2. Se **event ≠ LOGOFF, incremento = ZERO**. Pode ser uma sessão em que o logon no banco já tenha sido feito. Quando a conexão persiste por muito tempo, o valor de seus contadores já pode ter tido um incremento muito alto. Logo, considera-se apenas as diferenças (deltas) de valores subsequentes da sessão.
2. Se **sid == last_sid**: pode ser a mesma sessão ou o SID pode ter sido reaproveitado.
   1. Se **last_event == LOGOFF**, incremento = valor. Houve logoff daquele mesmo SID. Logo, o valor incrementado não pode ser delta, mas o valor atual.
   2. Se l**ast_event ≠ LOGOFF**, incremento = delta. Indica continuação de sessão identificada no registro anterior. Logo, o consumo é delta (value - lastval).

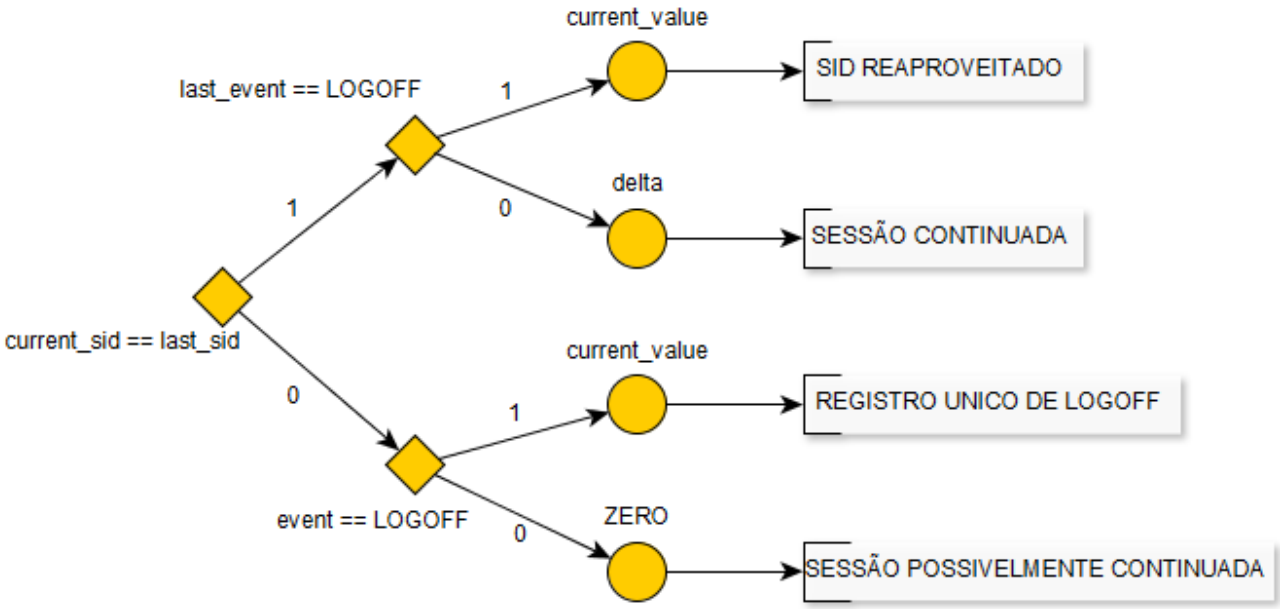


Figura 2: Condições de incremento

Para instalações feitas a partir do zero, em que a tabela TESTATISTICAS_RAC já possui o

atributo LOGON_TIME, a condição current_sid ≠ last_sid e event == LOGOFF não deve existir, pois a trigger de LOGON necessariamente será executada.

3.4 Condição de erro

A única condição em que as estatísticas podem ser processadas de forma errada é quando um determinado SID é reutilizado mais de uma vez no intervalo de um segundo. Neste caso, os atributos INST_ID, SID, SCHEMANAME e LOGON_TIME serão os mesmos e a ordenação interna das sessões irá falhar.

Uma forma de corrigir esse erro para instalações subsequentes é trocar os atributos TIME_LOGON de TESTATISTICAS_RAC para o tipo "TIMESTAMP", permitindo a distinção das sessões. No entanto, foi verificado que esta condição não ocorre com muita frequência e que o consumo de recursos dessas sessões curtas poderia ser desconsiderado. **Portanto, este modelo pode apresentar um desvio considerável quando a frequência dessas conexões muito curtas é alta**.

Um exemplo desse tipo de ocorrência pode ser vista a seguir. No caso abaixo, o usuário WIKIUSER faz logon e logoff em uma fração menor que um segundo. E ainda no mesmo segundo, o usuário USPL02 também faz um LOGON, fazendo logoff no segundo seguinte. Neste caso específico, a ordenação pelo parâmetro PROGRAM funcionaria, pois "http*" precede "SILEGIS". No entanto, se o programa, ao invés de começar com http, começasse com "Teste", a ordem abaixo seria mantida, pois o timestamp mais recente continuaria a ser do registro do SILEGIS. Logo, a ordenação última por programa não faria efeito.

prog	inst	sid	last_sid	schema	timestamp	val	lastval	delta	event	last_event	incr	acumulado	logon_time
SILEGIS-AQ-silegis	1	157	157	USPL02	2018-08-24 08:46:20	0	5	0	0	4	0	0	2018-08-24 08:46:20
httpd2-prefork@wiki (TNS V1-V3)	1	157	157	WIKIUSER	2018-08-24 08:46:20	0	0	0	0	0	0	0	2018-08-24 08:46:20
httpd2-prefork@wiki (TNS V1-V3)	1	157	157	WIKIUSER	2018-08-24 08:46:20	1	0	1	0	0	1	1	2018-08-24 08:46:20
httpd2-prefork@wiki (TNS V1-V3)	1	157	157	WIKIUSER	2018-08-24 08:46:20	2	1	1	4	0	1	2	2018-08-24 08:46:20
httpd2-prefork@wiki (TNS V1-V3)	1	157	157	WIKIUSER	2018-08-24 08:46:20	3	2	1	4	4	3	5	2018-08-24 08:46:20
SILEGIS-AQ-silegis	1	157	157	USPL02	2018-08-24 08:46:21	1	3	0	4	4	1	1	2018-08-24 08:46:20

4 Estatísticas Globais

Além das estatísticas de sessões, o Oracle também possui estatísticas globais, cujos valores podem ser obtidos de GV/V$SYSSTAT e GV/V$SYS_TIME_MODEL. No sistema de coleta de estatísticas do Oracle, optou-se por coletá-las periodicamente, através dos scripts roda-amostragem.sh e roda-amostragem.sql, descrito na Tabela 1.

Caso for necessário executar com periodicidade diferente, o usuário pode "detachar" os comandos SQL que alimentam as tabelas TESTATISTICAS_SYSSTAT e TESTATISTICAS_SYS_TIME_MODEL e executar um script similar em intervalos de tempo diferentes.

5 Instalação e checklist

1. Baixar o código-fonte (já vem com o script de instalação) no GIT (repositório **oracle-estatisticas-servidor**).

2. No arquivo **env.sh**, configurar todas as variáveis que são utilizadas para instalar os objetos (tabelas, procedures, triggers e scripts).
   1. É necessário ter o usuário e senha de DBA do Oracle (geralmente usuário SYS) para a criação do usuário e tablespace onde estarão os objetos de estatística;

3. Executar o arquivo **instala.sh** (que executa os demais scripts em ordem numérica).
   1. Se o usuário já existir, o instalador grava os objetos no mesmo schema. **A instalação irá falhar se o objeto que está sendo criado já existir no schema**, conforme item 4 da sessão 7 (Correções e melhorias).
   2. A execução e possíveis erros podem ser verificados no arquivo **instala.log**.

Ao término da execução do script, as tabelas, procedures e funções estarão instalados no schema definido pelo usuário e alguns itens de agendamento (crontab) serão criados:

```
# Execução do roda-amostragem.sh, de 10 em 10 minutos
3,13,23,33,43,53  7-19  * * *       $ORACLE_HOME/plus/admin/estatisticas/roda-amostragem.sh

# Execução do script de análise para geração de dados por programa e por sessão.
*/5  * * * *           $ORACLE_HOME/plus/admin/estatisticas/analisa_sessoes.sh

# Execução do roda-amostragem.sh, de 30 em 30 minutos, no período entre 20h e 07h.
00,30  20-23,00-06  * * *  $ORACLE_HOME/plus/admin/estatisticas/roda-amostragem.sh

# Execução do sumariza-amostragem.sh as 22:15. Deverá ficar obsoleto!
15 22 * * *           $ORACLE_HOME/plus/admin/estatisticas/sumariza-amostragem.sh

# Gera a tabela com os dados de consumo do dia, por programa e por hora.
*/10 * * * *           $ORACLE_HOME/plus/admin/estatisticas/sumariza-amostragem-perl.sh
CURRENT

# Gera a tabela HISTORICA com os dados de consumo do dia, por programa e por hora.
# Apenas dados entre 07:30 e 22:15 são coletados.
30 22 * * *           $ORACLE_HOME/plus/admin/estatisticas/sumariza-amostragem-perl.sh
WORKTIME

# Gera a tabela HISTORICA com os dados de consumo do dia, por programa e por hora.
# Os dados de todos os horários são coletados.
05 00 * * *           $ORACLE_HOME/plus/admin/estatisticas/sumariza-amostragem-perl.sh
FULLDAY
```

6 Visualização dos dados em ferramentas gráficas

Um dos maiores problemas era visualizar os dados em forma de relatórios. Para isso, foi utilizado o Apache Superset [9], uma ferramenta desenvolvida em Python e SQLAlchemy, através da qual é possível visualizar os dados armazenados em tabelas de banco de dados.

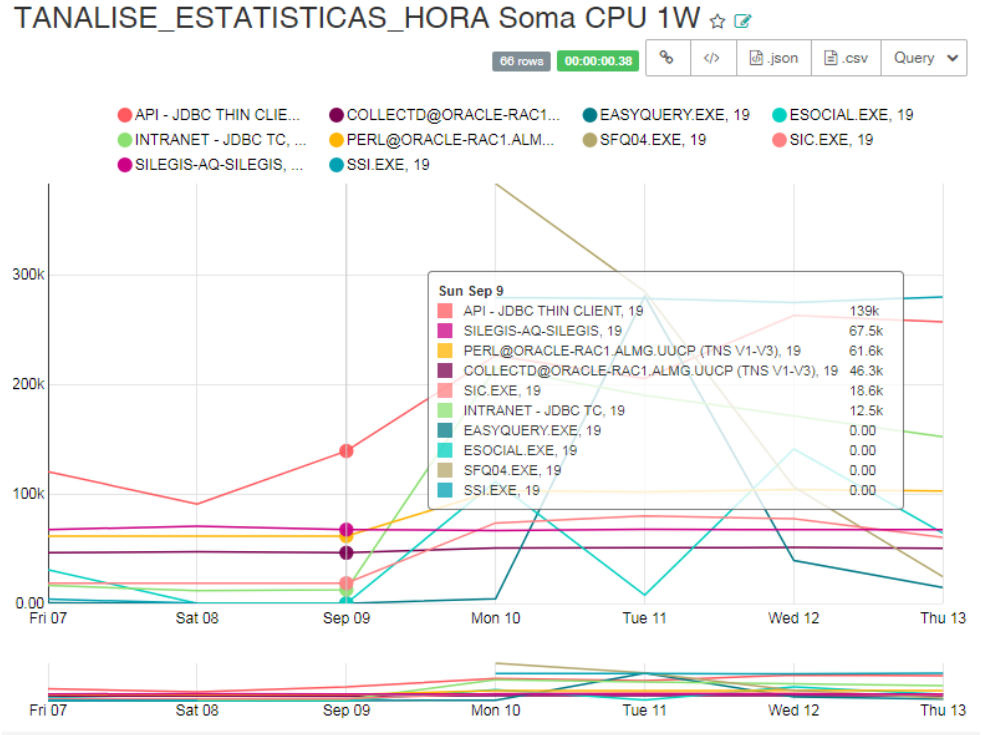
Através do superset é possível fazer várias combinações de visualização gráfica das tabelas geradas pelos processos descritos na Tabela 2. Alguns exemplos são mostrados a seguir:

- Tabela de consumo das sessões que utilizam mais CPU durante o dia (tabela TANALISE_ESTATISTICAS_SESSOES):

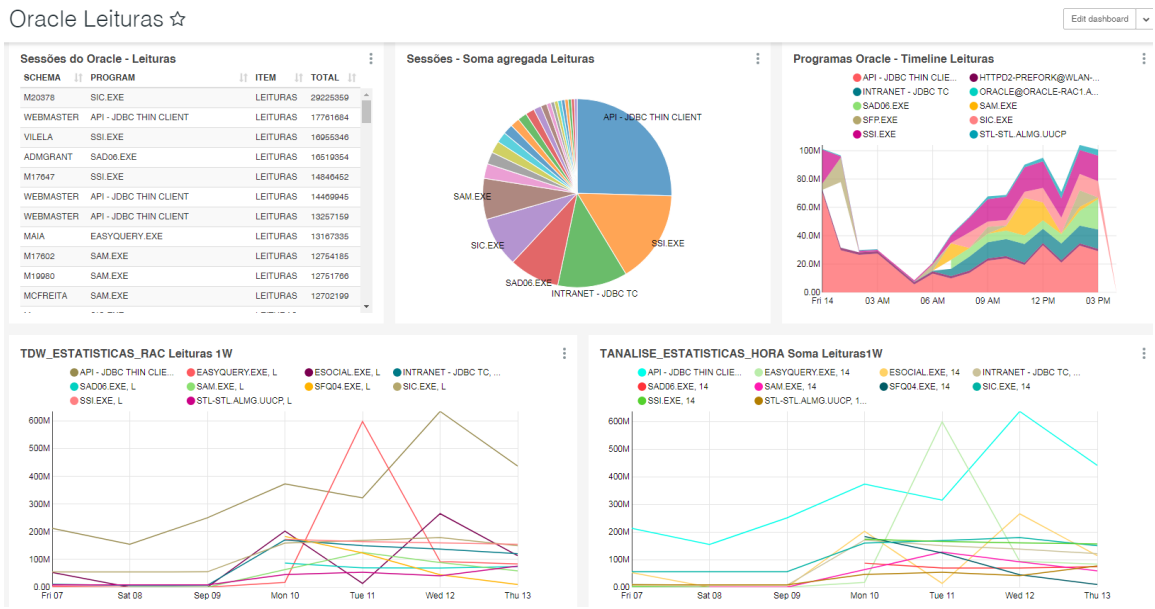
Sessões do Oracle - CPU ☆ 100 rows 00:00:00.37 .json .csv Query

SCHEMA	PROGRAM	ITEM	TOTAL
COLLECTD	COLLECTD@ORACLE-RAC1.ALMG.UUCP (TNS V1-V3)	CPU	29866
CHICOC	SMC.EXE	CPU	10782
VILELA	SSI.EXE	CPU	10522
WEBMASTER	API - JDBC THIN CLIENT	CPU	10357
M08450	SFQ04.EXE	CPU	9996
MAIA	ESOCIAL.EXE	CPU	9896
M25224	SSI.EXE	CPU	9667

- Gráfico do consumo de CPU histórico, por dia (tabela TANALISE_ESTATISTICAS_HORA com opção de "Time Grain" de 1 dia – o próprio superset já faz a agregação de todas as horas do dia):



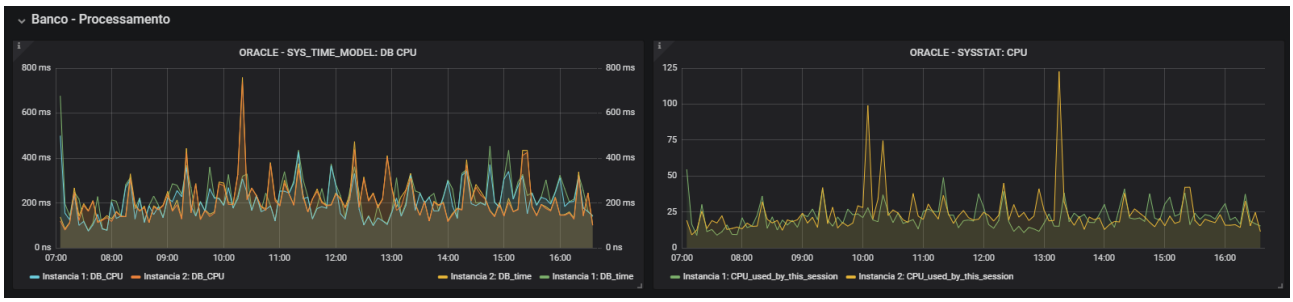
- É possível criar vários gráficos e juntar todos em um mesmo *dashboard* (ex: dashboard contendo a tabela com consumo de sessões, gráfico de pizza com as sessões agregadas por programa, gráfico de área por sessão e por hora e gráficos históricos).



Para a visualização de dados de estatísticas globais (views GV/V$SYSSTAT e GV/V$SYS_TIME_MODEL) optou-se pela utilização dos softwares collectd [10], graphite [11] e grafana [12]. O collectd é um software instalado no servidor Oracle (ou em outro servidor que tenha conexão com o Oracle) que executa queries e envia o resultado para o graphite, que armazena no formato whisper [13] (similar ao RRD). O grafana faz a solicitação desses dados ao graphite e apresenta em formato gráfico.

Um exemplo possível é configurar a seguinte query para rodar de 10 em 10 segundos, gerando o gráfico exibido abaixo:

```
SELECT INST_ID, NAME AS STAT_NAME, VALUE \
       FROM GV$SYSSTAT \
       WHERE STAT_ID IN (SELECT STAT_ID FROM TTIPO_ESTATISTICA_SYSSTAT)
```



7 Correções e melhorias

1. Alterar o atributo LOGON_TIME para o tipo TIMESTAMP e verificar se é possível corrigir sessões com início no mesmo segundo, conforme relatado em 3.4 (Condição de erro).
2. A partir do ano de 2019, uma vez que a estrutura da tabela TESTATISTICAS_RAC já possui o atributo LOGON_TIME, corrigir / adaptar o algoritmo de processamento descrito pela Figura 2.
3. Outras que vierem a surgir, de acordo com as demandas de desempenho do banco de dados.
4. Fazer uma verificação prévia de todos os nomes de objetos do schema onde os objetos de estatística serão criados. Caso algum objeto de mesmo nome seja encontrado, notificar o usuário que a instalação não será bem sucedida.

8 Apêndice A: métricas selecionadas

ID	Nome	Descrição	Fonte
1*	Requests to/from client	Não documentado. Provavelmente é a quantidade de requisições que chegam dos clientes para o servidor e vice-versa.	GV$SYSSTAT
14	session logical reads	"db block gets" + "consistent gets". Inclui leituras lógicas de blocos de dados da buffer cache ou memória privada.	GV$SYSSTAT
19	CPU used by this session	Utilização efetiva de CPU em N dezenas de milissegundos ou (N / 100) segundos.	GV$SYSSTAT
20	DB time	Tempo em centissegundos (cs) ou (N / 100) segundos.	GV$SYSSTAT
24	user I/O wait time	Tempo total N em 100s de espera de usuário por I/O (classe User I/O wait) ou n / 100 segundos.	GV$SYSSTAT
26*	non-idle wait time[1]	Não documentado. Provavelmente é o tempo médio de espera não ociosa por recursos (ex: espera por I/O ou por um row lock).	GV$SYSSTAT
100	db block changes	Alterações via update ou delete. Essas alterações geram registros de redo log e tornam-se permanentes caso o commit seja feito.	GV$SYSSTAT
230*	file io service time	Não documentado. Provavelmente é o tempo gasto com I/O para operações com arquivos.	GV$SYSSTAT
231*	file io wait time	Não documentado. Deve incluir o tempo de espera na fila + tempo de serviço.	GV$SYSSTAT
630	bytes sent via	Bytes enviados para os clientes. Indica volume de	GV$SYSSTAT

1 http://blog.orapub.com/20140805/what-is-oracle-db-time-db-cpu-wall-time-and-non-idle-wait-time.html

	SQL*Net to client	dados que trafega na rede do Oracle para os clientes (resultados de consultas).	
631	bytes received via SQL*Net from client	Bytes recebidos dos clientes. Indica tamanho + quantidade de consultas que o banco processa.	GV$SYSSTAT
633	bytes sent via SQL*Net to dblink	Bytes enviados via DB Link.	GV$SYSSTAT
634	bytes received via SQL*Net from dblink	Bytes recebidos via DB Link.	GV$SYSSTAT

* Indicador não documentado na página do Oracle.

3649082374	DB time	Atividade não ociosa do banco em microssegundos.	GV$SYS_TIME_MODEL
2748282437	DB CPU	Amount of CPU time (in microseconds) spent on database user-level calls. This does not include the CPU time spent on instance background processes such as PMON.	GV$SYS_TIME_MODEL
2821698184	sql execute elapsed time	Amount of elapsed time SQL statements are executing. Note that for select statements this also includes the amount of time spent performing fetches of query results.	GV$SYS_TIME_MODEL

9 Apêndice B: Oracle DB Time vs CPU used by this session

DB time: tempo do banco de dados. Contabiliza operações ociosas (como o sleep do banco!).
CPU used by this session: contabiliza apenas o tempo efetivo de CPU.
Ambas contabilizam o tempo em centissegundos (dividir por 100 para dar o tempo em segundos).

Exemplo da diferença entre DB Time e CPU used by this session[2]

```
set timing on
select name, m.value/100
from v$mystat m, v$sysstat s
where
  m.statistic# = s.statistic#
  and name IN ('DB time', 'CPU used by this session')
/
```

2http://international-dba.blogspot.com.br/2012/09/db-time-from-vsesstat.html

```
NAME                                                             M.VALUE/100
---------------------------------------------------------------- -----------
CPU used by this session                                              ,01
DB time                                                               ,03
Decorrido: 00:00:00.02
```

host sleep 10

```
/
NAME                                                             M.VALUE/100
---------------------------------------------------------------- -----------
CPU used by this session                                              ,01
DB time                                                               ,04
Decorrido: 00:00:00.00
```

exec dbms_lock.sleep(10) → gasta 10s de tempo não ocioso (para executar a chamada sleep!)

```
/
NAME                                                             M.VALUE/100
---------------------------------------------------------------- -----------
CPU used by this session                                              ,02
DB time                                                             11,13
Decorrido: 00:00:11.08
```

```
-- Bloco SQL anônimo utilizando apenas CPU (sem leitura ou escrita)
declare
  a number := 1;
  nloops number := 100000000;
begin
  for i in 1..nloops loop
    a := ( a + i )/11;
    --dbms_output.put_line(a);
  end loop;
end;
/

select name, m.value/100
from v$mystat m, v$sysstat s
where
  m.statistic# = s.statistic#
  and name IN ('DB time', 'CPU used by this session')
/
```

```
NAME                                                             M.VALUE/100
---------------------------------------------------------------- -----------
CPU used by this session                                            16,89
DB time                                                             28,02
Decorrido: 00:00:16.88
```

Documentação do STATSPACK ($ORACLE_HOME/rdbms/admin/spdoc.txt):

6.1. DB time compared to Total Call Time

DB time is a statistic manintained by the database, and was introduced in Oracle 10g Release 10.1. It can be queried in the view v$sys_time_model, along with other time model statistcs.

DB time measures the amount of time spent in the database by foreground processes in non-Idle state (e.g. the process is either on the CPU, or actively waiting for a resource or action). There is also a foreground CPU time statistic called 'DB CPU'.

Non-idle time spent in the database is also accumulated for background processes separately, in the time model statistic 'background elapsed time' (there is also a 'background cpu time').

Call Time, or Total Call Time is a proxy computed by Statspack for 'DB Time + bacground elapsed'. It is based on the v$sysstat statistic 'CPU used by this session' + 'time spent in non-Idle waits' (where Idle events are in stats$idle_event).

These numbers are usually very close, but may diverge. Either can be used to perform accurate problem diagnosis. Statspack uses both Call Time, and DB time, and AWR exclusively uses DB time.

10 Referências

1. Oracle Database Performance Tuning Guide / Automatic Performance Statistics (https://docs.oracle.com/cd/E11882_01/server.112/e41573/autostat.htm)
2. Diagnosing performance using STATSPACK (http://www.oracle.com/technetwork/database/performance/statspack-129989.pdf)
3. STATSPACK and AWR Statistics Comparison (http://www.remote-dba.net/oracle_10g_tuning/t_awr_statspack_statistics_comparison.htm)
4. Adjusting STATSPACK Collection levels (http://www.dba-oracle.com/tips_oracle_statspack_collection_levels.htm)
5. Oracle DBMS_APPLICATION_INFO (https://docs.oracle.com/cd/B28359_01/appdev.111/b28419/d_appinf.htm)
6. V$STATNAME (https://docs.oracle.com/cd/E18283_01/server.112/e17110/dynviews_3076.htm)
7. Oracle Statistics Descriptions (https://docs.oracle.com/cd/E18283_01/server.112/e17110/stats002.htm)
8. Oracle® Database 2 Day Developer's Guide, Cap 8, Using Triggers (https://docs.oracle.com/cd/E18283_01/appdev.112/e10766/tdddg_triggers.htm#insertedID1)
9. Apache (Incubator) Superset (https://github.com/apache/incubator-superset)
10. Collectd (https://collectd.org/)
11. Graphite App (https://graphiteapp.org/)
12. Grafana (https://grafana.com/)
13. Whisper database (https://graphite.readthedocs.io/en/latest/whisper.html)