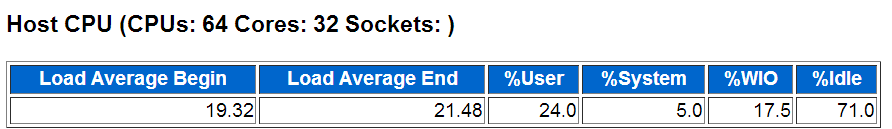
**>Host CPU**



**“Load Average” begin/end值代表CPU的大致运行队列大小。上图中快照开始到结束，平均 CPU负载增加了。**

**%User+%System=> 总的CPU使用率，在这里是29.0%**

**Elapsed Time \* NUM\_CPUS \* CPU utilization= 299.70 (mins) \* 64 \* 29%= 5562.4 mins=Busy Time <>idle time**

**与《Operating System Statistics》中的LOAD相呼应**

**host cpu反应整个主机的负载，并不是数据库的负载**

**I>nstance CPU**



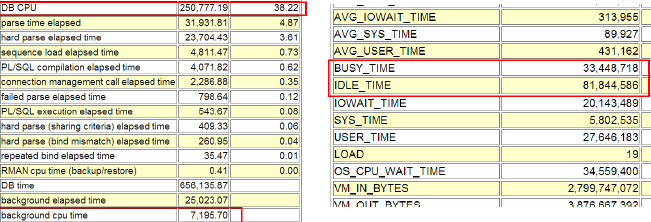
%Total CPU,该实例所使用的CPU占总CPU的比例 è % of total CPU for Instance   --有多少cpu和我的实例有关

%Busy CPU，该实例所使用的Cpu占总的被使用CPU的比例 è % of busy CPU for Instance

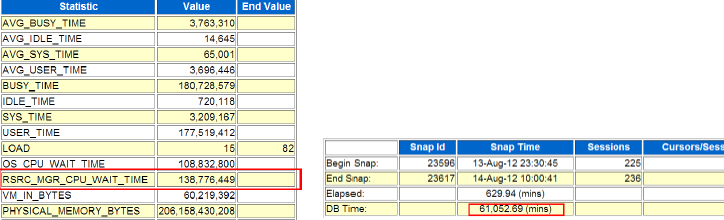
例如共4个逻辑CPU，其中3个被完全使用，3个中的1个完全被该实例使用，则%Total CPU= ¼ =25%，而%Busy CPU= 1/3= 33%

当CPU高时一般看%Busy CPU可以确定CPU到底是否是本实例消耗的，还是主机上其他程序

% of busy CPU for Instance= （DB CPU+ background cpu time) / (BUSY\_TIME /100)= (250,777.19 + 7,195.70)/ (33,448,718/100)= 77.1%

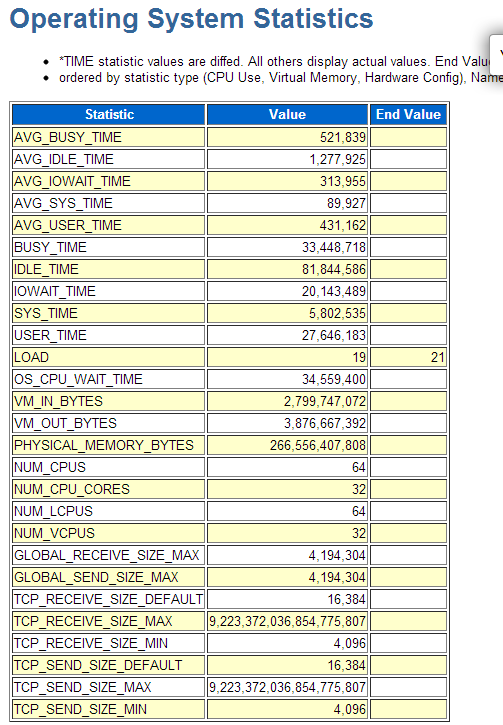


%DB time waiting for CPU (Resource Manager)= (RSRC\_MGR\_CPU\_WAIT\_TIME/100)/DB TIME= (138,776,449/100)/(61,052 \* 60)= 37.88%



resmgr:cpu quantum等待事件是当resource manager 控制CPU 调度时，需要控制对应进程暂时不使用 CPU而进程到内部运行队列中，以保证该进程对应的 consumer group(消费组 )没有消耗比指定 resource manager指令更多的 CPU。

**>Operating System Statistics**



数据来源于V$OSSTATèDBA\_HIST\_OSSTAT，, TIME相关的指标单位均为百分之一秒

Busy\_Time=SYS\_TIME+USER\_TIME

AVG\_BUSY\_TIME= BUSY\_TIME/NUM\_CPUS

BUSY\_TIME + IDLE\_TIME = ELAPSED\_TIME \* CPU\_COUNT=299.7\*60\*64=1150848s=(33,448,718+81,844,586)/100

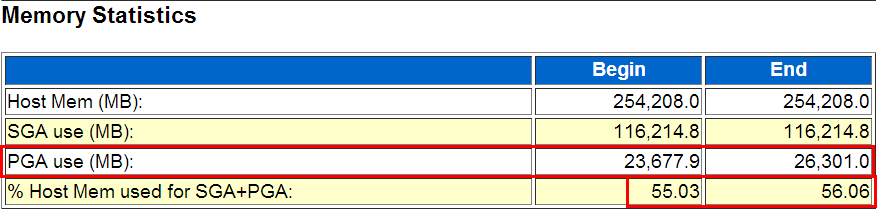
OS\_CPU\_WAIT\_TIME è进程等OS调度，cpu queuing

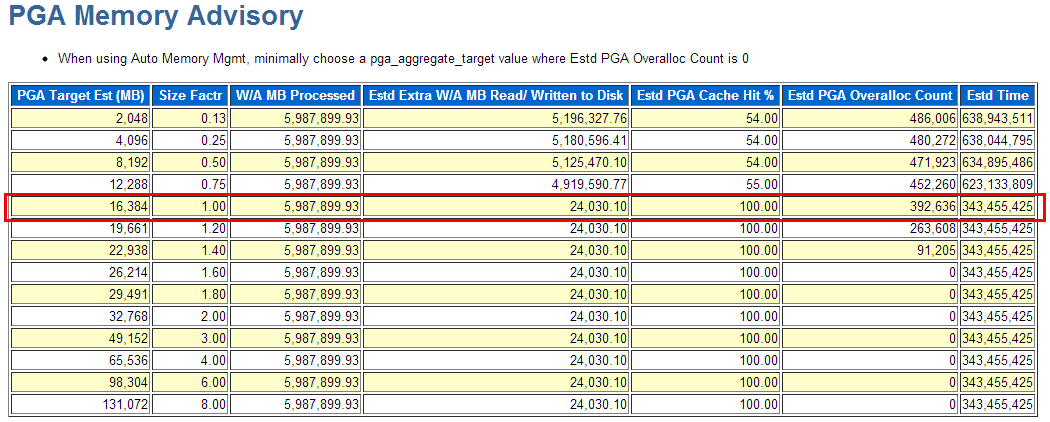
VM\_IN\_BYTES 2.6GB PGAE IN

VM\_OUT\_BYTES 3.6GB PGAE OUT è部分版本下并不准确，例如Bug 11712010 Abstract: VIRTUAL MEMORY PAGING ON 11.2.0.2 DATABASES，仅供参考

**还是需要借助oswatcher 和nmon工具去判断**

>Memory Statistics





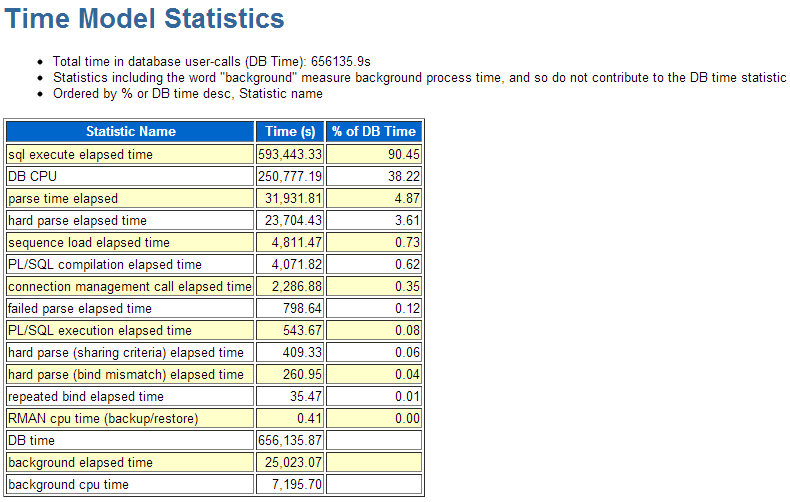
11g以后才有的一个section，可以了解SGA、PGA、主机物理内存的大致使用情况。

如上例PGA在快照时间内从23677MB增长到26301MB, pga\_aggregate\_target=16GB，存在overalloc

% Host Mem used for SGA+PGA 可以大致反映本实例占用主机物理内存的情况。

注意Host Mem也可能起伏，如使用DLPAR时

**>Time Model Statistics**



Important statistics here is the DB Time. The statistic represents total time spent in database calls. It is calculated by aggregating the CPU time and wait time of all sessions not waiting on idle event (non-idle user sessions). Since this timing is cumulative time for all non-idle sessions, it is possible that the time will exceed the actual wall clock time.

Objective of tuning oracle database is to reduce the amount of time users spend in performing some action on the database.

This time represents time taken by foreground sessions and not background sessions.

In above example, DB Time is 49,209.52 seconds. 91% of the time is being spent on “SQL execution elapsed time” i.e for SQL execution.

DB CPU represents time spent on CPU resource by foreground user processes. This time doesn’t include waiting time for CPU. DB CPU is contributing to 22% of total DB time.

Important thing to note here is, the actual wall clock time is around 3600 seconds (difference between 2 snapshots) but the DB CPU shown here is 10,932 seconds.

If the server machine (on which database server is running) has more than 1 CPUs, it is possible to have DB CPU greater than actual wall clock time. In this example the database server is hosted on machine with 8 CPUs. Hence a 1 second is divided into 8 parts. Hence DB CPU 10,932 seconds means, 10,932 (DB CPU) / 8 (CPU) = 1366 wall clock seconds.

The “Parse time elapsed” and “Hard parse elapsed time” has taken around 17% and 15% of the total DB time. “Parse time elapsed” represents time spent for Syntax and Semantic checks. Whereas “Hard parse include time” represents time spent for Syntax and Semantic checks PLUS time spent for optimizing the SQL and generating optimizer plan.

Time Model Statistics几个特别有用的时间指标：

• parse time elapsed、hard parse elapsed time 结合起来看解析是否是主要矛盾，若是则重点是软解析还是硬解析

• sequence load elapsed time sequence序列争用是否是问题焦点

• PL/SQL compilation elapsed time PL/SQL对象编译

• 注意PL/SQL execution elapsed time 纯耗费在PL/SQL解释器上的时间。不包括花在执行和解析其包含SQL上的时间

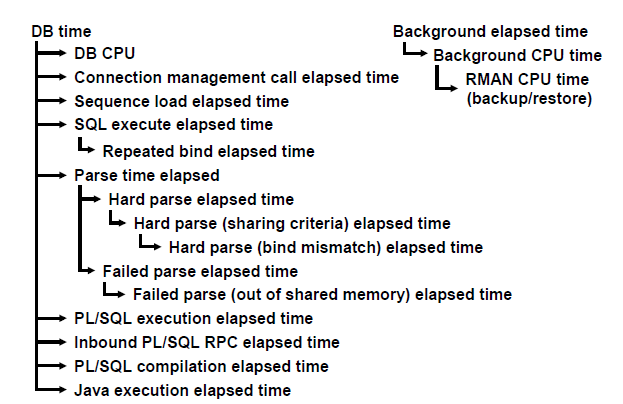
• connection management call elapsed time 建立数据库session连接和断开

• failed parse elapsed time 解析失败，例如由于ORA-4031

• hard parse (sharing criteria) elapsed time 由于无法共享游标造成的硬解析

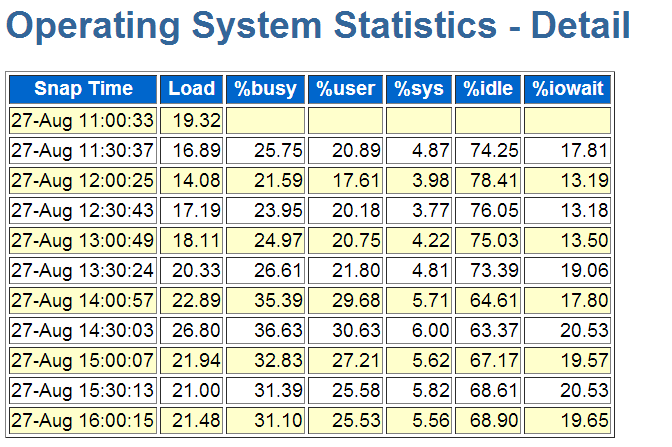
• hard parse (bind mismatch) elapsed time 由于bind type or bind size 不一致造成的硬解析

time modle statistics 树形图



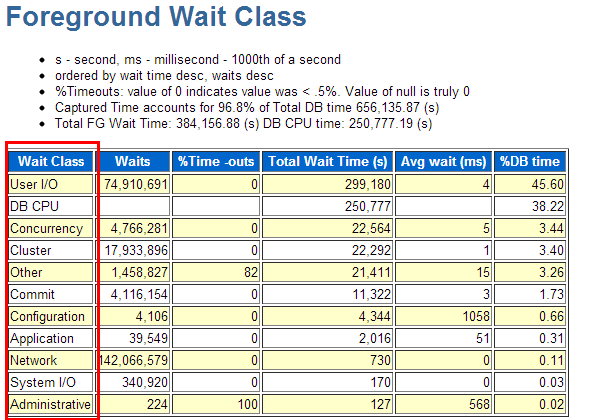
存在包含关系所以 Time Model Statistics加起来超过 100%再正常不过

>Operating System Statistics - Detail



每个快照对应一行记录，简易了解OS负载情况 针对CPU Spike毛刺问题有点用，也可看做大致的cpu走势 虽然替代不了nmon、osw，但胜在AWR自带。 特别是对哪些没任何os性能监控的”处女环境”来说是个宝。

**>Wait Class**



The events are sorted in descending order of “Total Wait Time (s)” column.

There are over 800 distinct wait events. Oracle has grouped these wait events in 12 wait classes. These wait classes are further divided in 2 categories, Administrative Wait Class and Application Wait Class.

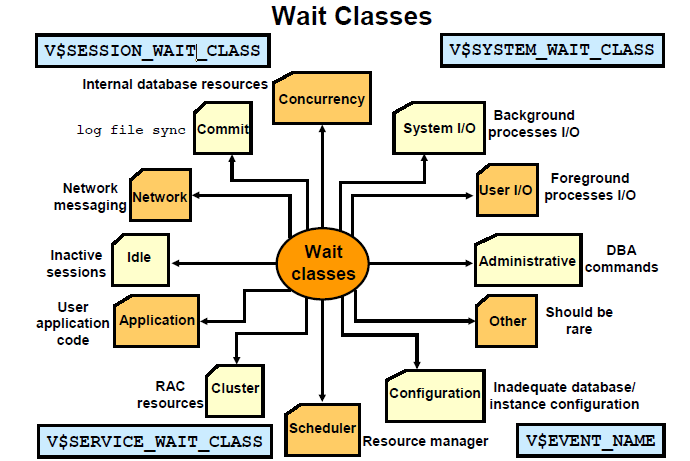
These wait classes gives overall information about whether the waits happening for Application or for System events.

In the example above first 2 rows show that total wait time is higher for Concurrency and “User I/O” wait classes. Though we won’t have much control on reducing concurrency, but we could aim at reducing the User IO.

High User IO means,

* From the pool of available indexes proper indexes are not being used
* FTS is happening on big tables with millions of rows

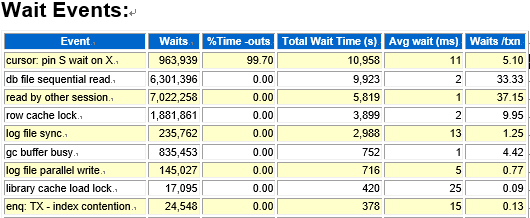
This could give pointers for further tuning.



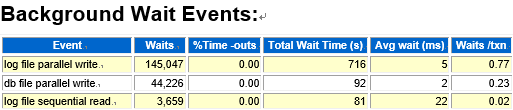
将等待分类

select distinct wait\_class from v$event\_name;

select name,wait\_class from v$event\_name order by 2;



The wait events are sorted on “Total Wait Time (s)” column in descending order. The idle events are listed down in the end. First 10 to 15 events should be looked into because rest of the events are idle events and can be ignore. These events are related to foreground processes.



The wait events are sorted on “Total Wait Time (s)” column in descending order. The idle events are listed down in the end. First 10 to 15 events should be looked into because rest of the events are idle events and can be ignore. These events are related to background processes.