ORACLE®

# Average active sessions: the magic metric?

John Beresniewicz
Consulting Member of Technical Staff
Oracle USA

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.

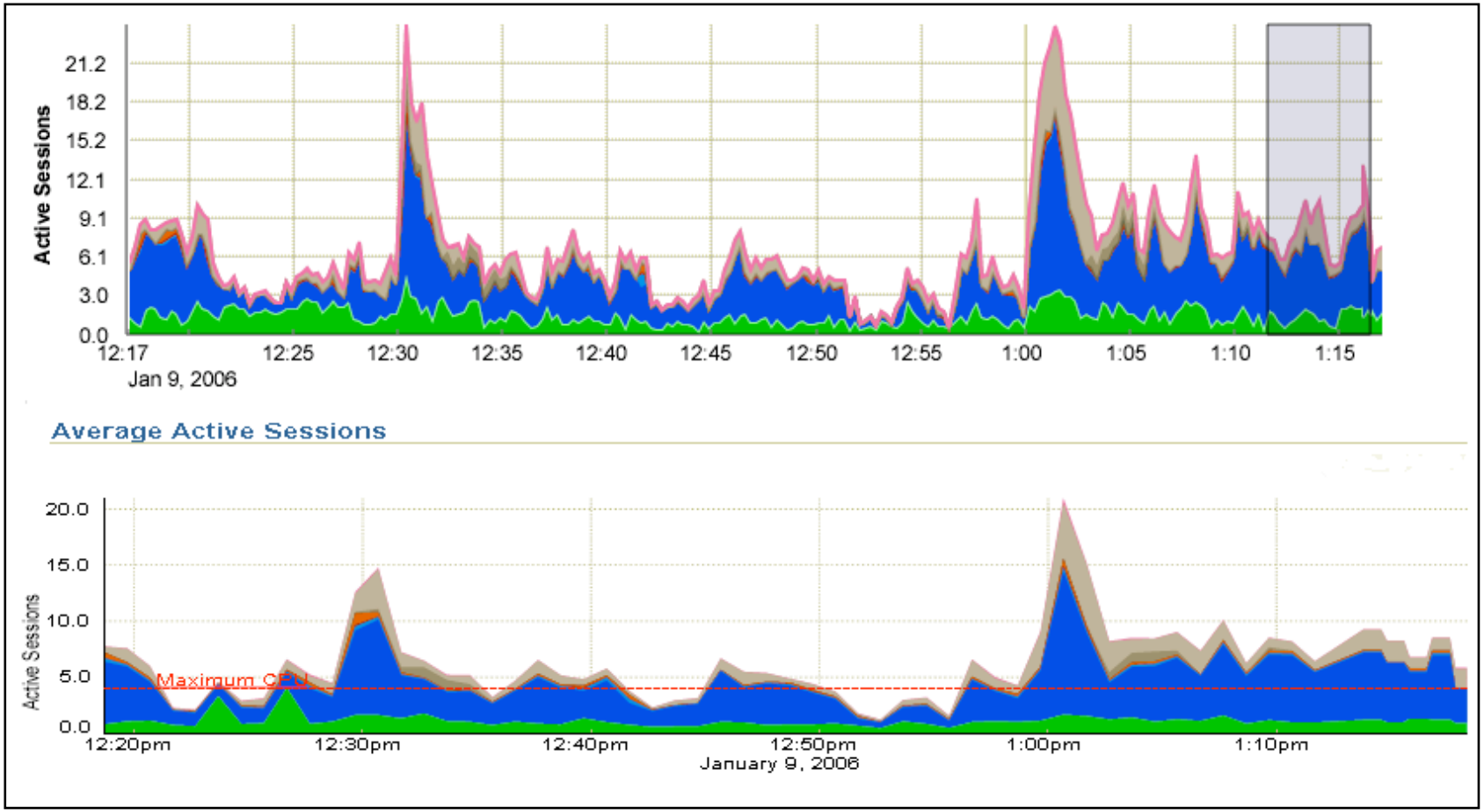
The development, release, and timing of any features or functionality described for Oracle's products remain at the sole discretion of Oracle.

# Topics entertained

- Database time and active sessions
- Average active sessions
- Oracle 10g V$ and AWR visibility
- Estimating DB time with ASH
- EM Performance Pages

- Appendix A: The calculus of DB time
- Appendix B: Little's Law and average active sessions

# What is this?


Active Sessions
21.2
18.2
15.2
12.1
9.1
6.1
3.0
0.0
12:17
12:25
12:30
12:35
12:40
12:45
12:50
12:55
1:00
1:05
1:10
1:15
Jan 9, 2006
Average Active Sessions
Active Sessions
20.0
15.0
10.0
5.0
0.0
Maximum CPU
12:20pm
12:30pm
12:40pm
12:50pm
1:00pm
1:10pm
January 9, 2006

# Database time

- Time spent in the database by foreground sessions
- Includes CPU time, IO time and wait time
- Excludes idle wait time
- The lingua franca for Oracle performance analysis

***Database time is total time spent by user processes either actively working or actively waiting in a database call.***

# Where is DB time used?

- AWR and AWR compare periods reports
- EM Performance page and drill downs
- ASH report
- Top SQL impact rankings for SQL Tuning Advisor
- Server-generated Alerts
  - See especially metric baselines and adaptive thresholds

# DB time counters

- Time Model
  - V$SYS_TIME_MODEL and DBA_HIST_SYS_TIME_MODEL
  - Stat_name = ‘DB time’
  - Values in microseconds
- Wait Model
  - V$WAITCLASSMETRIC_HISTORY and DBA_HIST_WAITCLASSMET_HISTORY
  - Columns DBTIME_IN_WAIT and TIME_WAITED
  - Values in centiseconds

# System load and DB time

- More users
  - => More calls
    - => DB time increases
- Larger transactions
  - => Longer calls
    - => DB time increases

***DB time increases as system load increases.***

# System performance and DB time

- IO performance degrades
  - => IO time increases
    - => DB time increases
- Application performance degrades
  - => Wait time increases
    - => DB time increases

***DB time increases as system performance degrades.***

# Active sessions

- Foreground sessions in a database call
  - Backgrounds are also interesting
- Either on CPU, waiting for IO, or waiting (not idle)
- V$ACTIVE_SESSION_HISTORY is a collection of timed regular samples of active session attributes

***Active sessions are foreground sessions contributing to DB time in any given moment.***

# Average active sessions

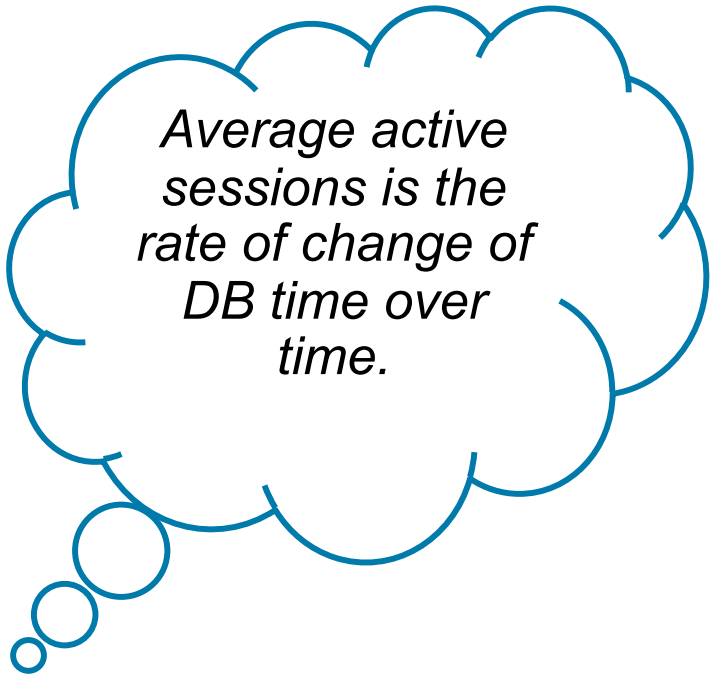
**AAS = DB time / elapsed time**

**(during some workload)**

- NOTE: Synchronize time units in numerator and denominator

# Average active sessions

- Time-normalized DB time
- Full-time equivalent sessions
  - Not whole sessions
  - How many full-time virtual sessions to do the work?
- Comparable
  - Across systems
  - Across time periods



# What are the units?

- Time / time = unitless?
- DB time accumulates in micro- or centi-seconds
- Time-normalized metrics are per second of elapsed

- Centi-seconds (foreground time) per second (elapsed)
- Centi-users per second
- User seconds per elapsed second (normalize time units)
- Active session seconds per second
- Active sessions

# V$ visibility (10g)

- V$SYS_TIME_MODEL
  - STAT_NAME = ‘DB time’
  - Accumulated value over entire instance
- V$WAITCLASSMETRIC_HISTORY
  - AVERAGE_WAITER_COUNT
    - It is precisely Average Active Sessions
- V$SYSMETRIC_HISTORY
  - “Database Time Per Second”, “CPU Usage Per Sec”
  - Units are Centi-seconds per second
  - Value is 100 x Average Active Sessions

# AWR visibility (10g)

- DBA_HIST_SYS_TIME_MODEL
- DBA_HIST_WAITCLASSMET_HISTORY
- DBA_HIST_SYSMETRIC_HISTORY
  - Enabling EM metric baselines persists 15 metrics to AWR

# AWR snaphots

- One hour snapshot interval is a great default
  - Diurnal periodicity in workloads (e.g. corporate email)
  - Attempts snapshots on whole hour
    - Simplifies hour-of-day analysis
- 7-day snapshot retention is a conservative default
  - 35 days for monthly/weekly cycles is nice for 10g R2 metric baselines and adaptive thresholds

```
DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS
    (interval  => 60       -- minutes
    ,retention => 20160    -- minutes
    )
```

# AWR average active sessions

- DBA_HIST_SYS_TIME_MODEL
  - Stat_name = ‘DB time’
  - Values in microseconds
- DBA_HIST_SNAPSHOT
  - Elapsed time = end_interval_time – begin_interval_time
  - Partition by startup_time (for instance)

***We need to divide DB time by wall clock time.***

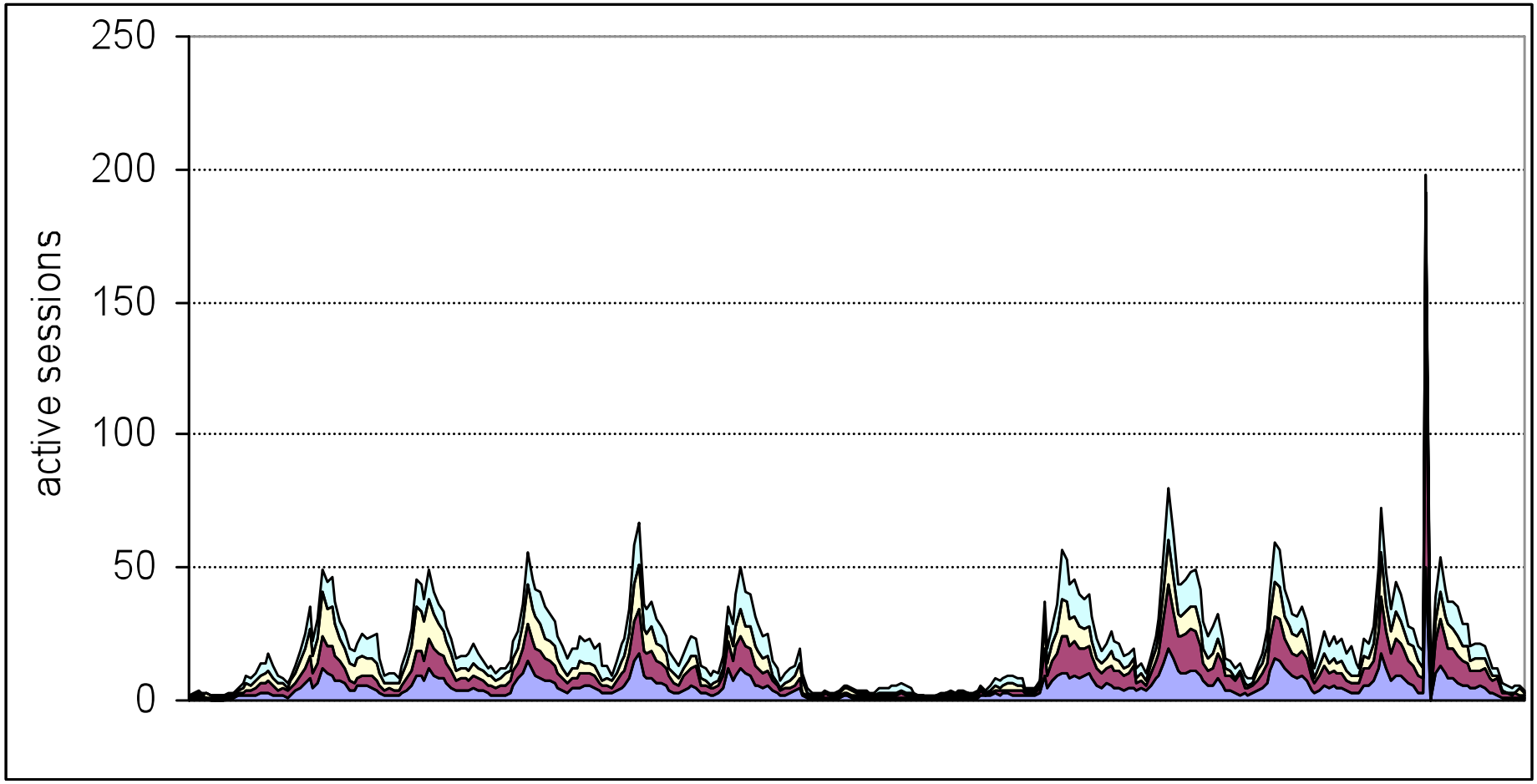
# Compute AWR average active sessions by snapshot

- Step 1: Prepare raw data stream
  - Join:
    DBA_HIST_SNAPSHOT
    DBA_HIST_SYS_TIME_MODEL
- Step 2: Compute elapsed and DB time deltas by snapshot
- Step 3: Compute average active sessions
  - DB time(delta) over elapsed time(delta)

# Compute AWR average active sessions

SQL\AWRsnapAAS.sql

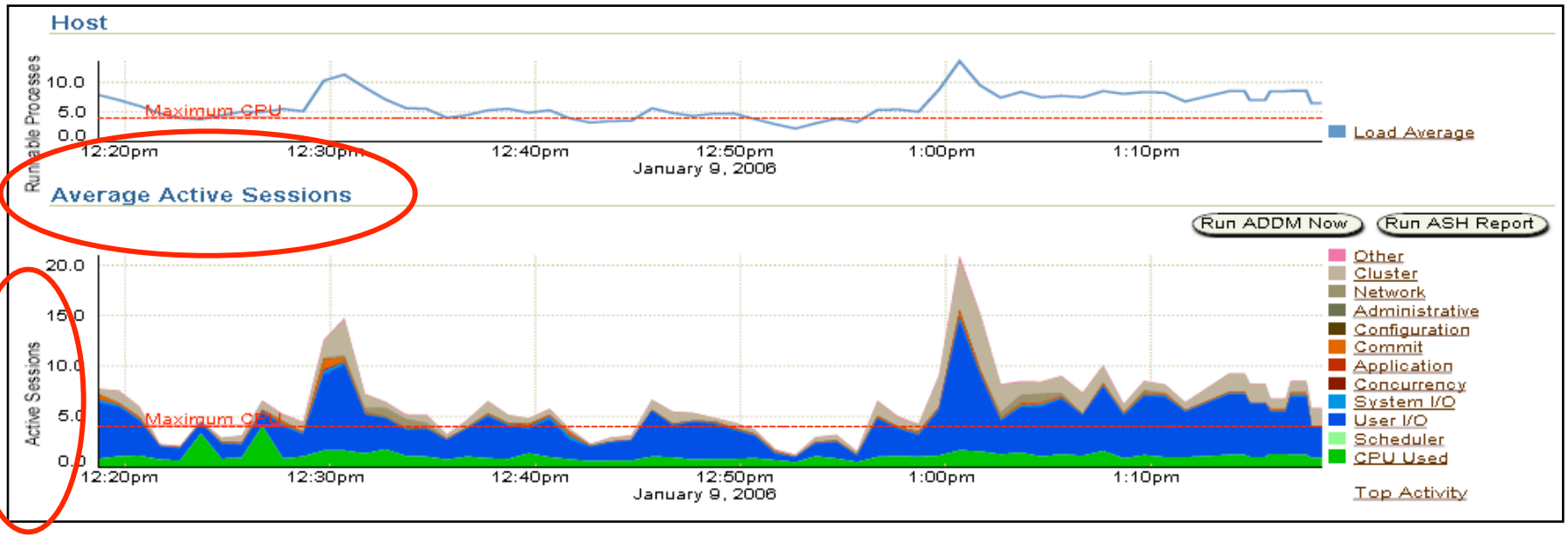
# Average active sessions: RAC email



# Performance monitoring

- Avg active sessions captures load and performance
  - Severe performance degradation can spike the metric
- Server-generated alert metrics for monitoring:
  - *Database Time Per Sec* (10g)
  - *Average Active Sessions* (11g)
- Server-set alert threshold values:
  - Set threshold values to statistically unusual (spike)
  - Compute statistics over baseline time periods
  - 10gR2 Metric Baselines ; 11g Adaptive Thresholds

# EM Performance page

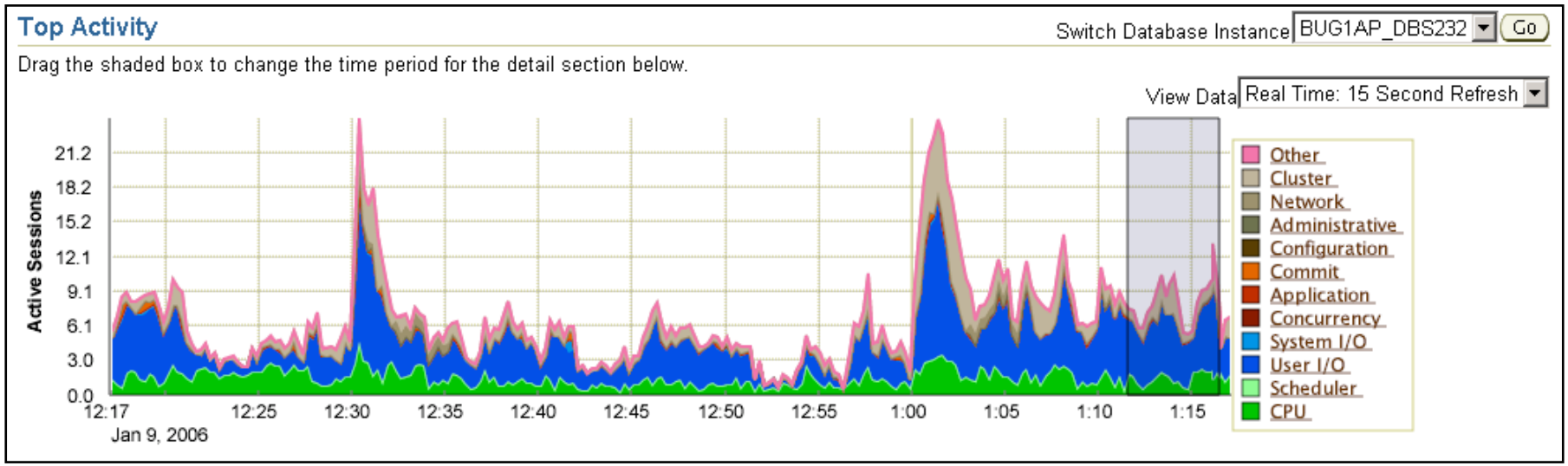



- Cumulative DB time by wait class
- v$waitclassmetric_history and v$sysmetric_history
- 1 minute intervals

# Active Session History (ASH)

- Persisted samples of active session information
  - Sessions contributing to DB time at time of sampling
- One-second sampling interval is a great default
  - Allows simplified AAS computations
- DB time and Average active sessions can be computed by aggregating ASH samples
  - See Appendix A: The calculus of DB time

# EM Top Activity page



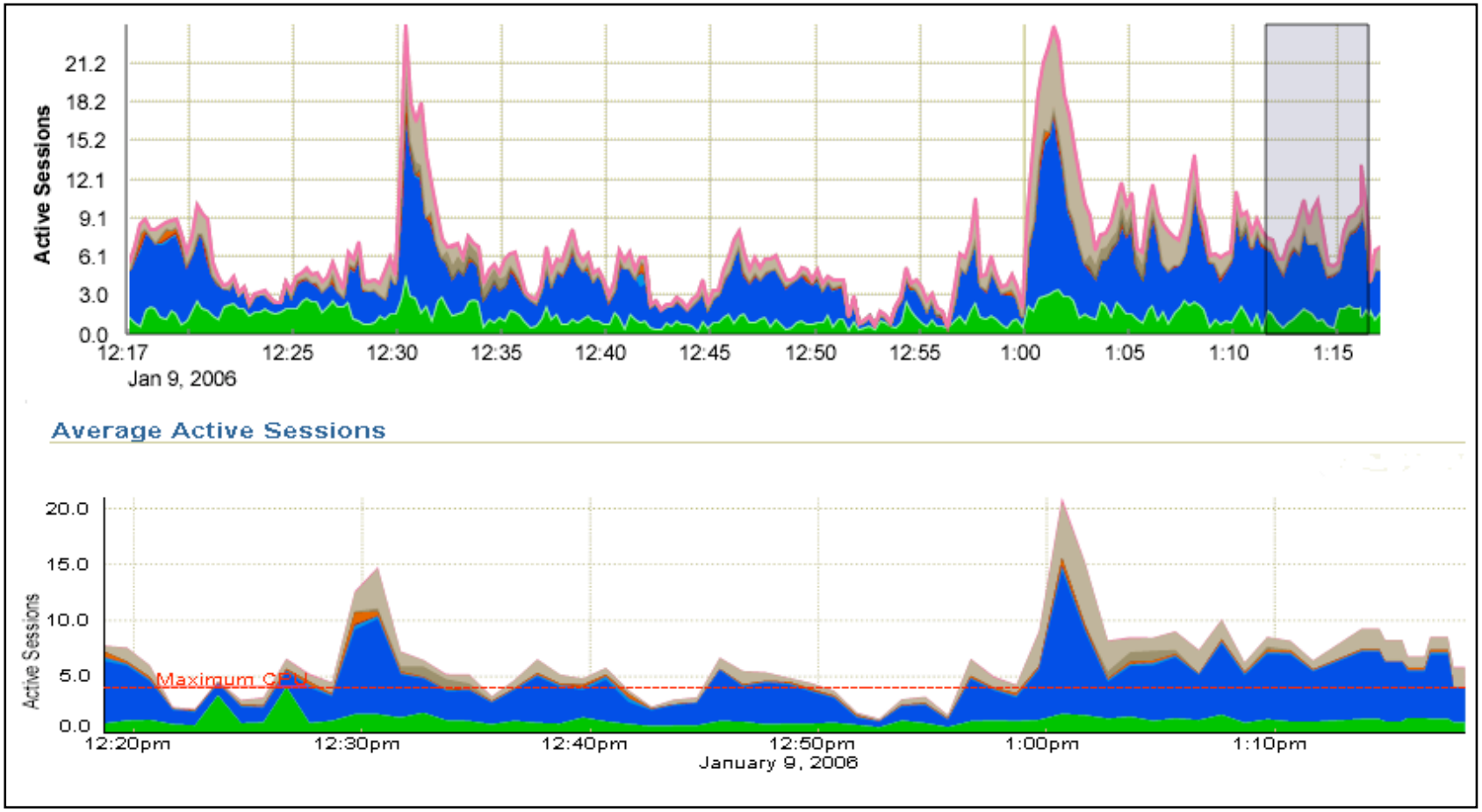

- ASH-estimated DB time by wait class
- Aggregated over 15 second intervals
- Thanks Kyle!

# ASH and DB time

- ASH sample counts = DB Time in seconds
  - Assumes default 1 second sampling
  - Low sample sizes are less reliable
- Look for skew within dimensions of interest
  - Sqlid, session id, module, instance

```
select COUNT(1) as dbtime
      ,sqlid
 from v$active_session_history
where session_type='FOREGROUND'
group by sqlid
order by 1;
```

# Estimated vs. cumulative DB time




***See Appendix A for details on why this works.***

# Compare ASH estimates to SYSMETRIC counters of DB time

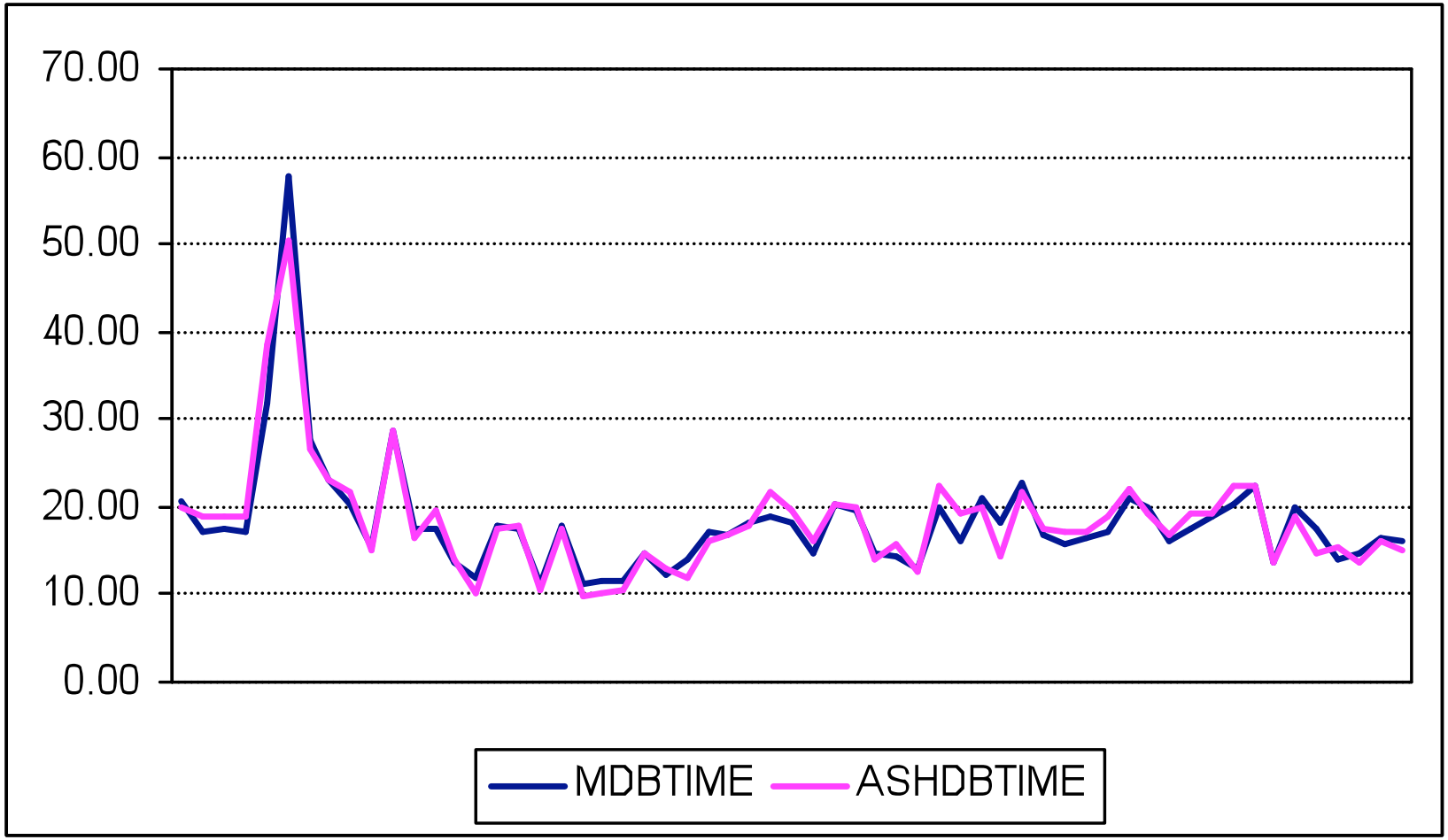
- Accumulated: V$SYSMETRIC_HISTORY
  - “Database Time Per Sec”
- Estimated: V$ACTIVE_SESSION_HISTORY
  - DB Time = sample counts
- Join by ASH sample time between SYSMETRICS begin_time and end_time

***Are there substantial differences between accumulated and estimated values of DB time?***

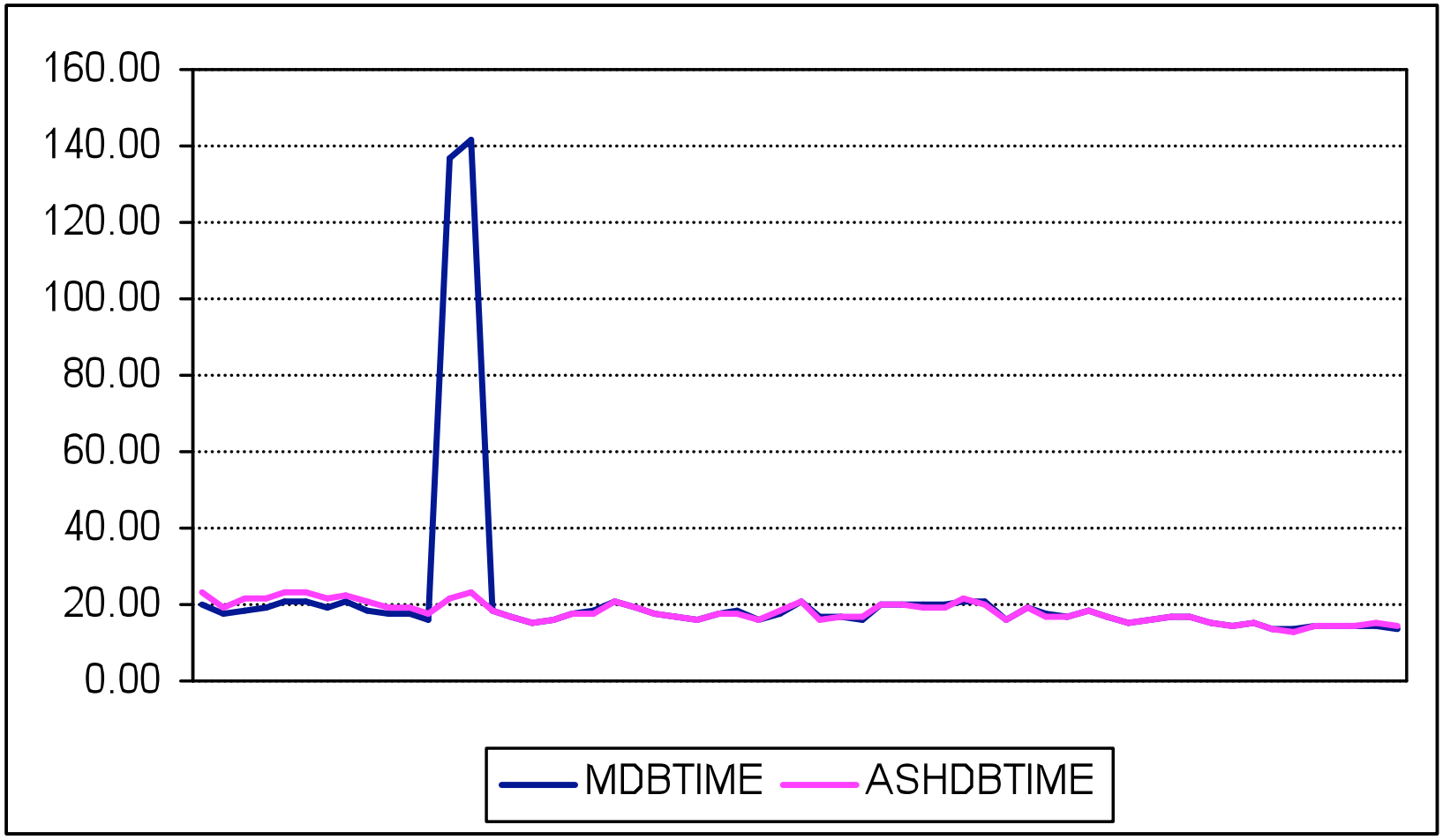
# Compare ASH estimates to SYSMETRIC counters of DB time

SQL\ASHmetricsAAS.sql

# Comparison results: email DB



# Comparison results: apps DB

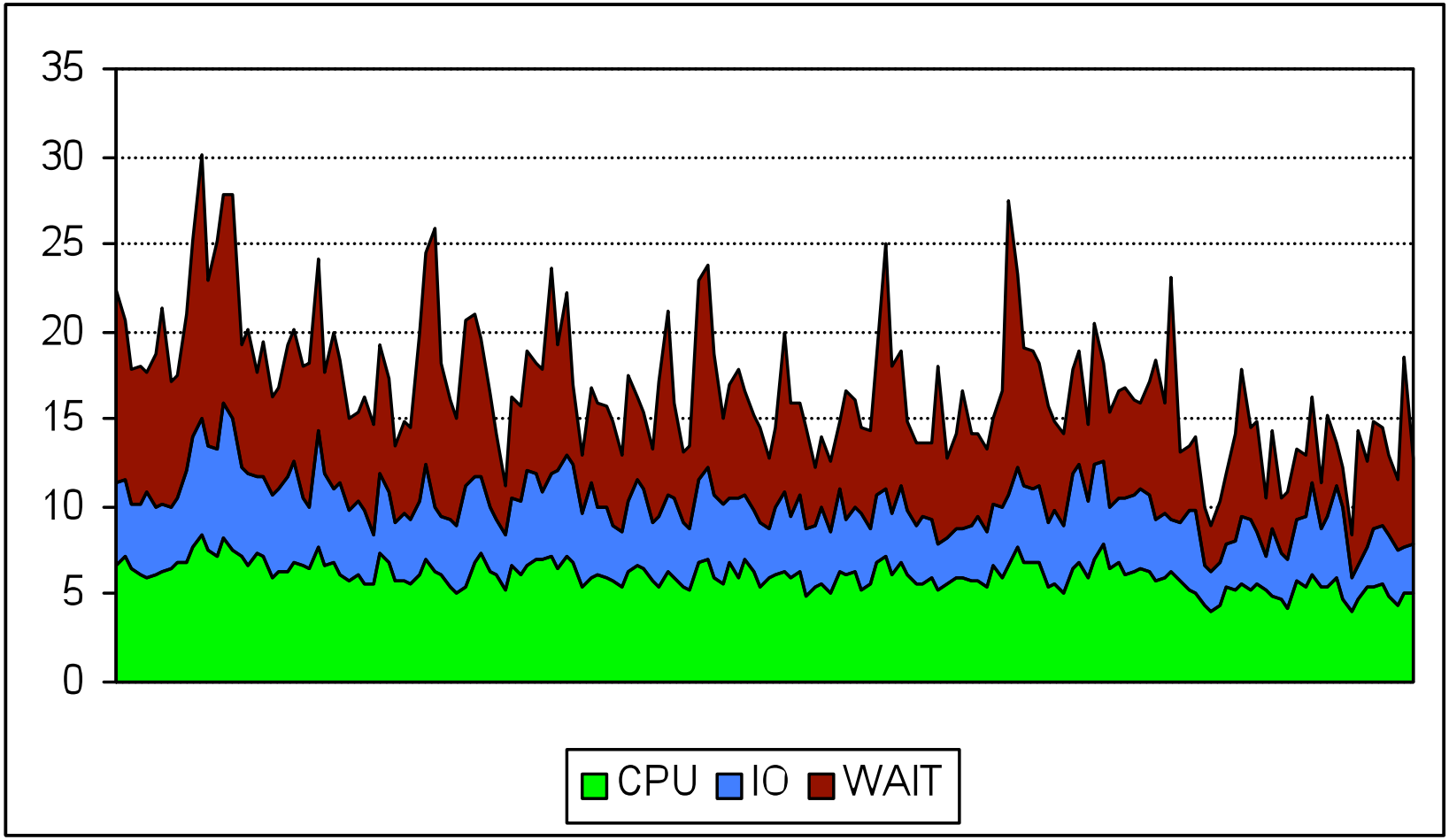



***Accumulation delays for long-running operations?***

# Decomposing DB time: CPU, IO, Wait

SQL\ASHcpuiowaitAS.sql

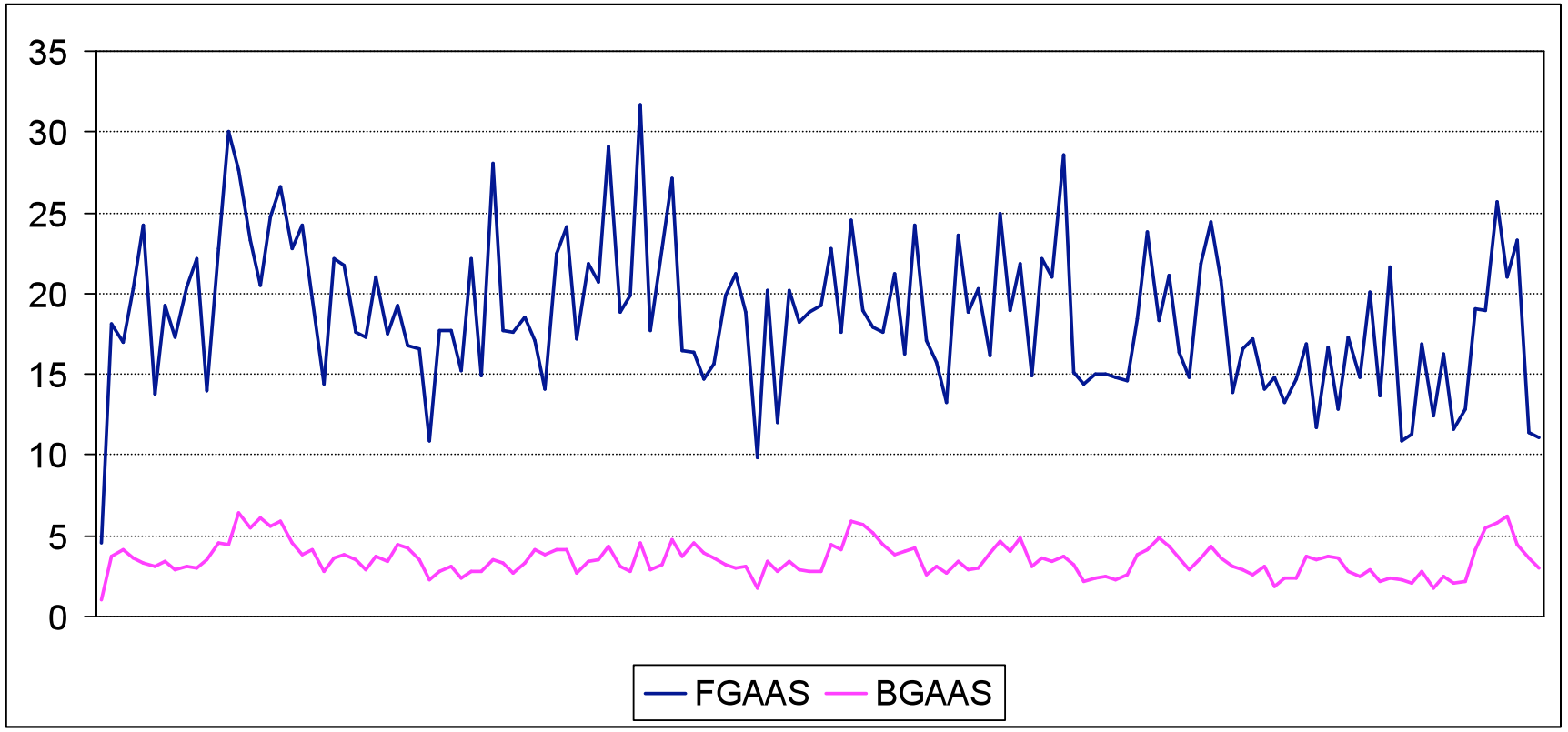
# Decomposing DB time: CPU, IO, Wait



# Analyze ASH dump: FG vs. BG activity

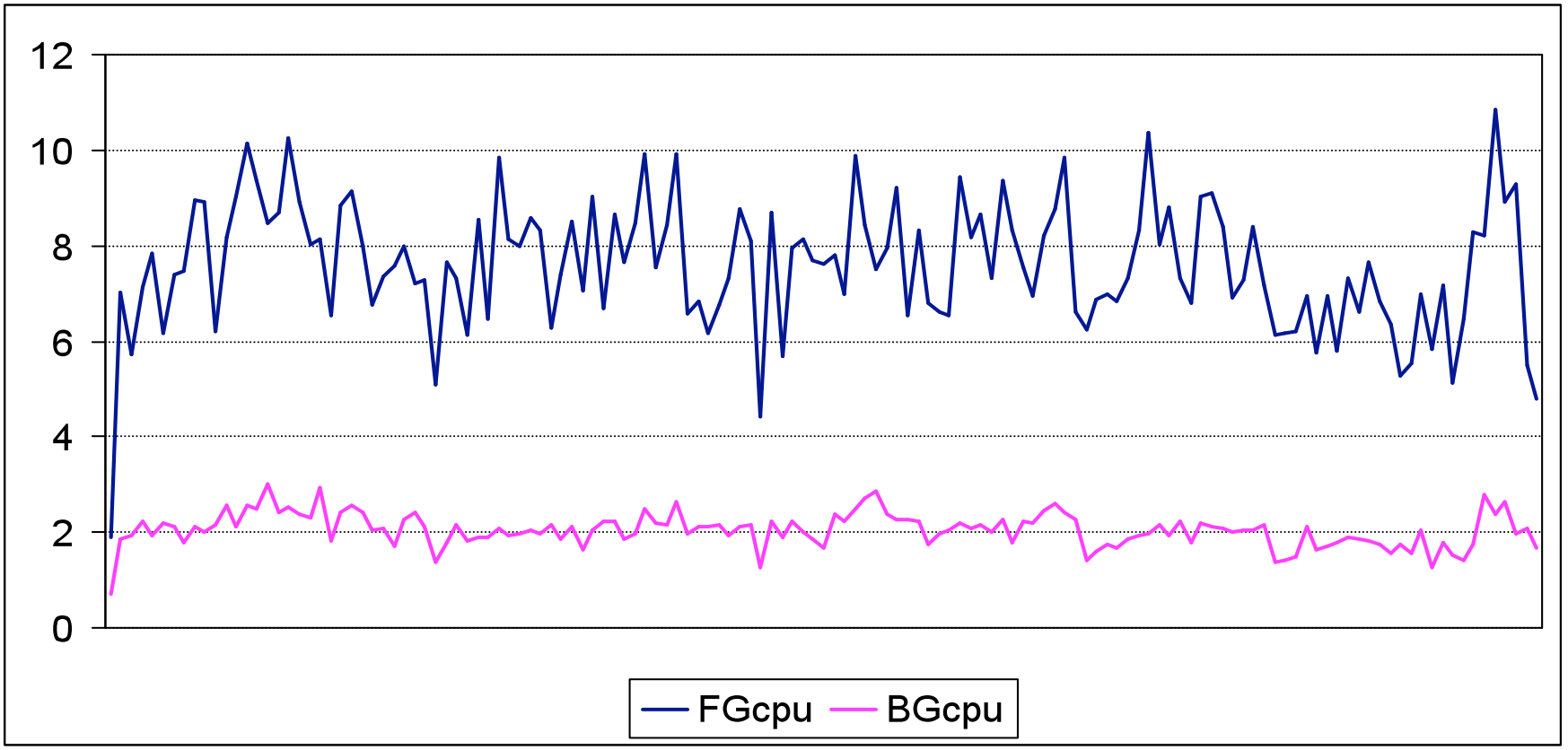
SQL\ASHdumpAAS.sql

# ASH dump:
# FB/BG avg active sessions




***80-20 split of FOREGROUND to BACKGROUND time on this system.***

# ASH dump:
# FG/BG avg active sessions on CPU




***This is a 20 CPU machine so there is headroom.***

# BG CPU time

```
select program
       ,count(*) as DBtime
  from gmamocs_ashdata
 where session_type=2
   and instance_number=1
 group by program having count(*) > 60
 order by 2
/
```

RAC LMS processes consuming significant CPU resources

| PROGRAM | DBTIME |
|---|---|
| oracle@rgmdbs1 (CJQ0) | 73 |
| gmdbs1 (CKPT) | 80 |
| gmdbs1 (MMON) | 126 |
| gmdbs1 (LCK0) | 129 |
| gmdbs1 (LMON) | 152 |
| gmdbs1 (m000) | 309 |
| gmdbs1 (m001) | 881 |
| gmdbs1 (DBW2) | 1089 |
| gmdbs1 (ARC0) | 1117 |
| gmdbs1 (SMON) | 1118 |
| gmdbs1 (DBW1) | 1123 |
| oracle@rgmdbs1 (LMD0) | 1313 |
| oracle@rgmdbs1 (ARC1) | 1325 |
| oracle@rgmdbs1 (DBW0) | 1341 |
| oracle@rgmdbs1 (**LMS3**) | **2745** |
| oracle@rgmdbs1 (**LMS4**) | **2746** |
| oracle@rgmdbs1 (**LMS1**) | **2824** |
| oracle@rgmdbs1 (**LMS2**) | **3070** |
| oracle@rgmdbs1 (**LMS0**) | **3146** |
| oracle@rgmdbs1 (LGWR) | 4239 |

# Summary

- DB time is the fundamental performance metric
  - Increases with both client load and degraded system performance
- Average active sessions is rate of change of DB time over time
  - Instantaneous load/performance indicator
- Oracle 10g Active Session History can be used to aggregate DB time across many useful dimensions
- Data-mining ASH for real-time indicators as well as historical trends or model input parameters is useful


ORACLE IS THE INFORMATION COMPANY

# Appendix A
# The calculus of DB time

ASH sample counting as a Riemann integral estimate of DB time

# The calculus of DB time

- The number of active sessions at any time is the rate of change of the DB time function at that time.
- DB time is the integral of the Active Session function.

$$DBtime(t_1) - DBtime(t_0) = \int_{t_0}^{t_1} ActiveSessions(t) * dt$$

# The DB time integral

$$\int_{t_0}^{t_1} ActiveSessions(t) * dt$$

$$= \lim \Delta t \to 0 \sum_{k=1}^{n} ActiveSessions(t_k) * \Delta t$$

$$(\text{where } n = (t_1 - t_0)/\Delta t)$$

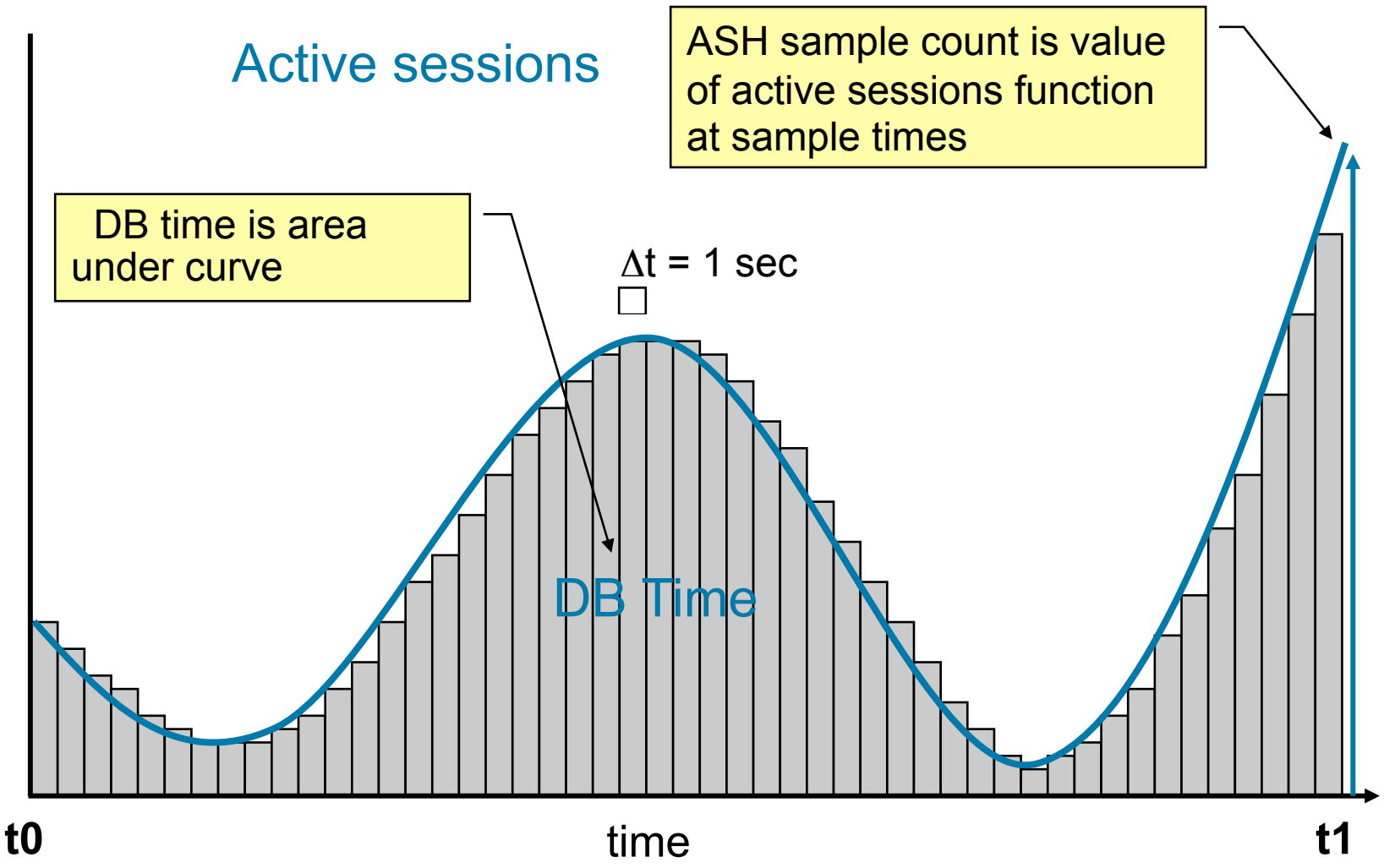
# Integral approximation using ASH

$$\lim \Delta t \to 0 \sum_{k=1}^{n} ActiveSessions(t_k) * \Delta t$$

$$\approx \sum_{sample time \geq t0}^{sample time \leq t1} ASH samples * 1$$

(where $\Delta t = 1$ second)

# Estimating DB time with ASH



# Appendix B
# Little's Law and average active sessions

The relationship of average active sessions to black-box queuing models.

# Little's Law for queuing systems

$$N = X * R$$

N = number of active requests in system

X = serviced request throughput

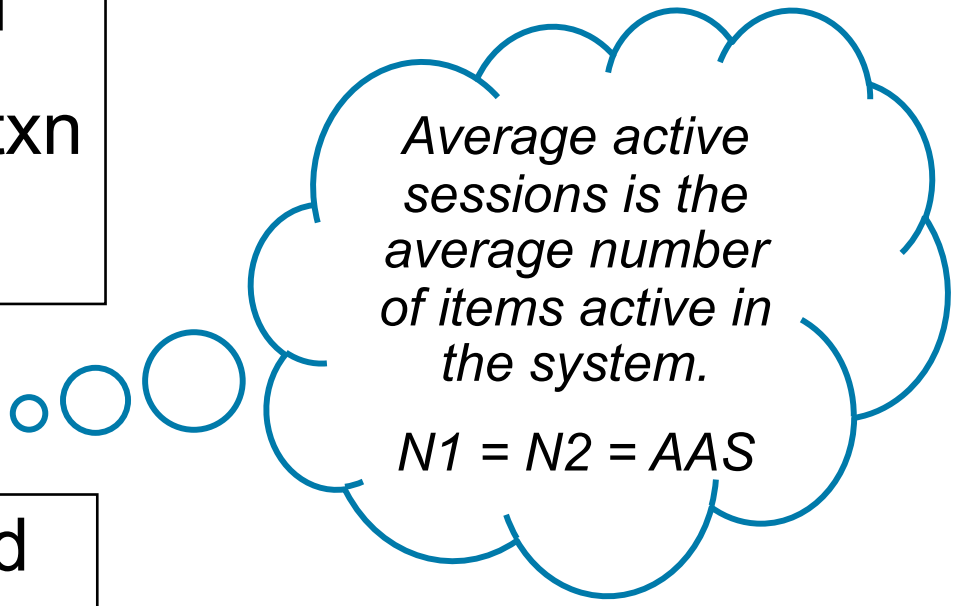
R = avg. service time per request

# Little's Law and SYSMETRICS

X1 = Txn per second

R1 = Response per txn

N1 = X1 * R1 = ?




X2 = Calls per second

R2 = Response per call

N2 = X2 * R2 = ?

# Little's Law and SYSMETRICS

Average active sessions

= Response per Call * Total calls per sec

= Response per Txn * Txn per sec

***This explains why DB time increases with both performance degradation and load increase.***

# Little's Law and SYSMETRICS

## SQL\LittlesLawAAS.sql

- These equivalences have been measured to 6 places on very busy systems
- Demonstrates the integrity of 10g instrumentation:
  - DB time used for response time computations is identical
  - Transaction and call counters are consistently updated


ORACLE®