

*SQL\*Net Message From Client* is your top contributor to response time – now what?

---

*Gary Goodman*  
*Hotsos Enterprises, Ltd.*

## Overview

---

- What is the *SQL \*Net message from client* event?
- Origin of the 'Idle Time' label
- The Hotsos Players
- Examples
  - Issue
  - Remedy
- Close

## Defining *SQL\*Net message from client*

---

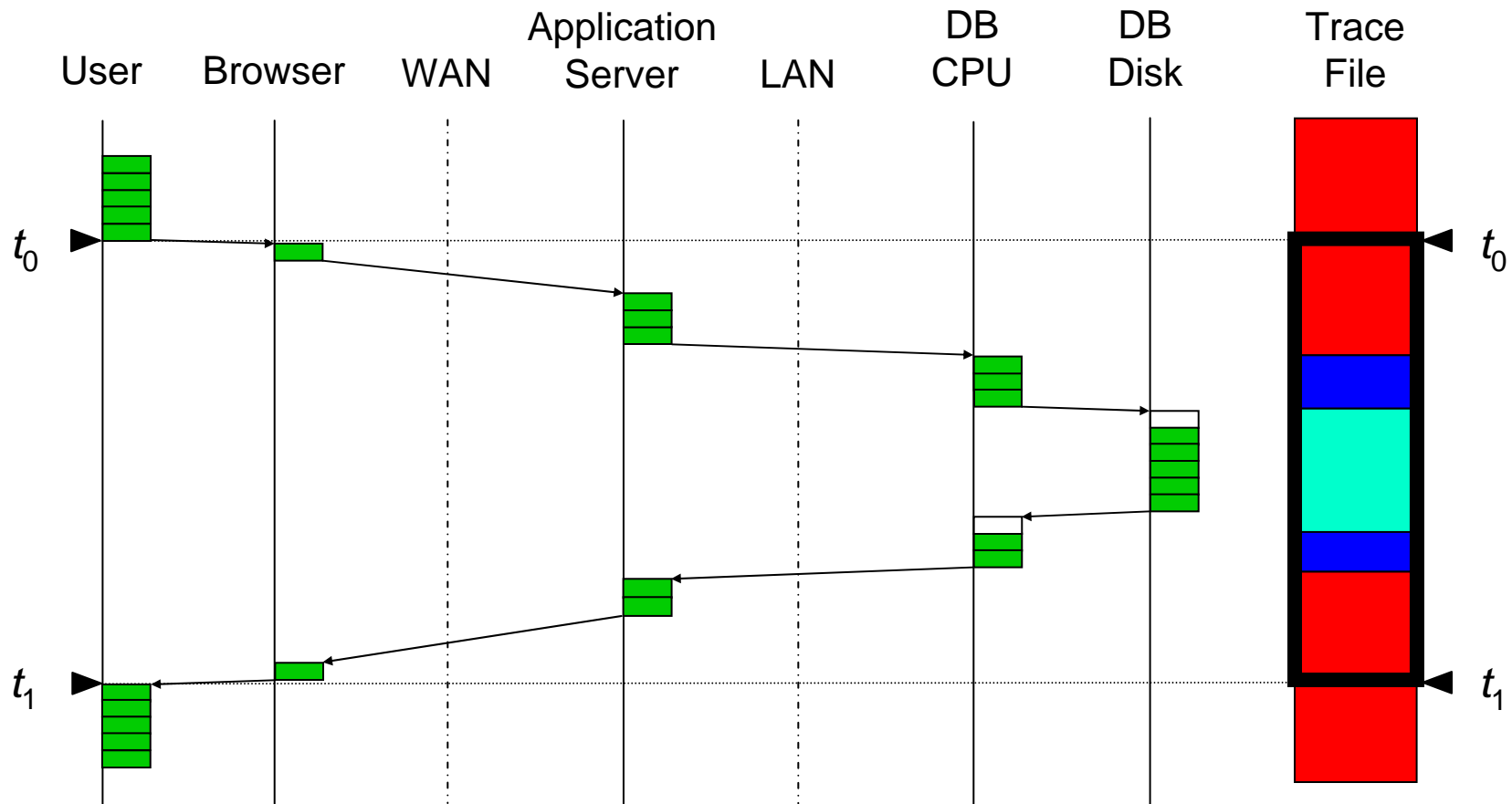
- Oracle waiting for a properly formatted database call
- To the OS it is a *read* call
  - read(7,
- Time between database calls
- OS gettimeofday time stamps
  - Set timed\_statistics = true
  - Event 10046 level 8 or 12

## The Hotsos Players

---

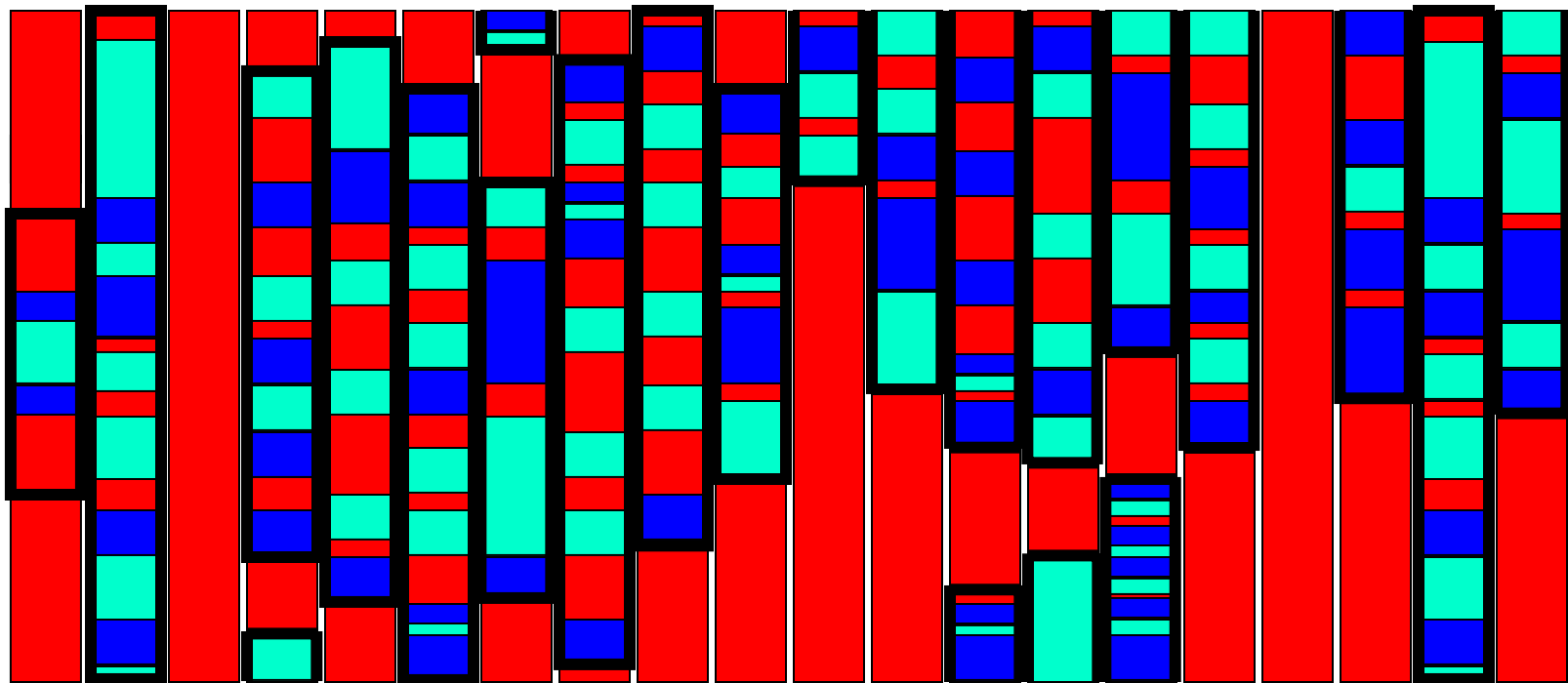
- Official Timers
  - Response time
  - *SQL\*Net message from client*
- “You are here”
- Browser
- Application server
- Oracle
  - CPU
  - Disk

Here's a picture of what we just watched.



In system-wide performance metrics, it's impossible to distinguish response time from "idle" time.

---



## The origin of the 'idle time' label

---

- *“We also have to filter out wait events that are not helpful to our tuning effort. The list below shows all system "idle" wait events that have no meaningful information:*
  - ...
  - *SQL\*Net message from client*
  - ...
- *“The next prerequisite to using bottleneck analysis is that certain wait events should be filtered out of any metrics used to diagnose performance bottlenecks. For example, Oracle will record a wait statistic that represents how long a particular user sits at their SQL\*Plus prompt between each issued database request.”*

## Root cause issues and remedies

---

- High count
  - The fastest way to do something is don't do it!
- High average duration
  - If you must do it, make each one faster

## High count: Parsing within a loop

---

- SQL from batch job

```
INSERT INTO STAGING_AREA (
    DOC_OBJ_ID,
    TRADE_NAME_ID,
    LANGUAGE_CODE,
    OBJECT_RESULT,
    GRAPHIC_FLAG,
    USER_LAST_UPDT,
    TMSP_LAST_UPDT)
VALUES (
    1000346,
    54213,
    'ENGLISH',
    '<BLANK>',
    'N',
    'sa',
    TO_DATE('11/05/2001 16:40:54', 'MM/DD/YYYY HH24:MI:SS')
)
```

## High count: Parsing within a loop

---

Cursor Action	Library Misses	Action Count	Rows Processed
Parse	348	348	0
Execute	0	348	348
Fetch	0	0	0
Total	348	696	348

- Each *parse* call is for a 'unique' statement
- Total database calls = 696
- *SQL\*Net message from client* events = 696!

## High count: Step 1 - Use bind variables!

---

```
INSERT INTO STAGING_AREA (
    DOC_OBJ_ID,
    TRADE_NAME_ID,
    LANGUAGE_CODE,
    OBJECT_RESULT,
    GRAPHIC_FLAG,
    USER_LAST_UPDT,
    TMSP_LAST_UPDT)
VALUES (
    :A1,
    :B1,
    :C1,
    :D1,
    :E1,
    :F1,
    TO_DATE( :G1, 'MM/DD/YYYY HH24:MI:SS' )
)
```

## High count: Step 2 – Move the *parse* call

---

- Parse once outside of the loop
- Execute inside a loop multiple times
  
- But to accomplish that....you need to complete step 1
  - Bind variables give you the OPPORTUNITY to share
  
- “A *soft parse* is how Oracle handles a parse call it should have never seen”

– Cary Millsap

## High count: Results of parsing once by using bind variables!

---

Cursor Action	Library Misses	Action Count	Rows Processed
Parse	1	1	0
Execute	0	348	348
Fetch	0	0	0
Total	1	349	348

- The ONLY *parse* call is for the statement with bind variables
- Total database calls = 349 (down from 696)
- *SQL\*Net message from client* events = 349 (down from 696)!

## High count: No array processing

---

Cursor Action	Library Misses	Action Count	Rows Processed
Parse	1	1	0
Execute	0	348	348
Fetch	0	0	0
Total	1	349	348

- Same code from before
- Processing one row with each execution (348 each)
- Batch job so there is no user interaction

## High count: Add array processing

---

Cursor Action	Library Misses	Action Count	Rows Processed
Parse	1	1	0
Execute	0	4	348
Fetch	0	0	0
Total	1	5	348

- Array processing of 100 rows per execution
- Database call count drops to 5 (from 349)
- *SQL\*Net message from client* events drop to 5 (from 349)!

## Final analysis -

---

- Before

Cursor Action	Library Misses	Action Count	Rows Processed
Parse	348	348	0
Execute	0	348	348
Fetch	0	0	0
Total	348	<b>696</b>	348

- After

Cursor Action	Library Misses	Action Count	Rows Processed
Parse	1	1	0
Execute	0	4	348
Fetch	0	0	0
Total	1	<b>5</b>	348

## High count: Twin parses

---

Cursor Action	Library Misses	Action Count	Rows Processed
Parse	303	606	0
Execute	0	303	303
Fetch	0	0	0
Total	303	909	303

- Why parse twice for every execute?
  - Parse
  - Parse again
  - Execute

## High count: Turn *describe* off!

---

Cursor Action	Library Misses	Action Count	Rows Processed
Parse	303	303	0
Execute	0	303	303
Fetch	0	0	0
Total	303	606	303

- Development tool option
  - Describe before parse for error handling
- Good during development – horrible during use!
- We have seen with both PowerBuilder and Perl
  
- Question: What else should we look at to continue optimizing?

## High average duration: tnsnames.ora alias

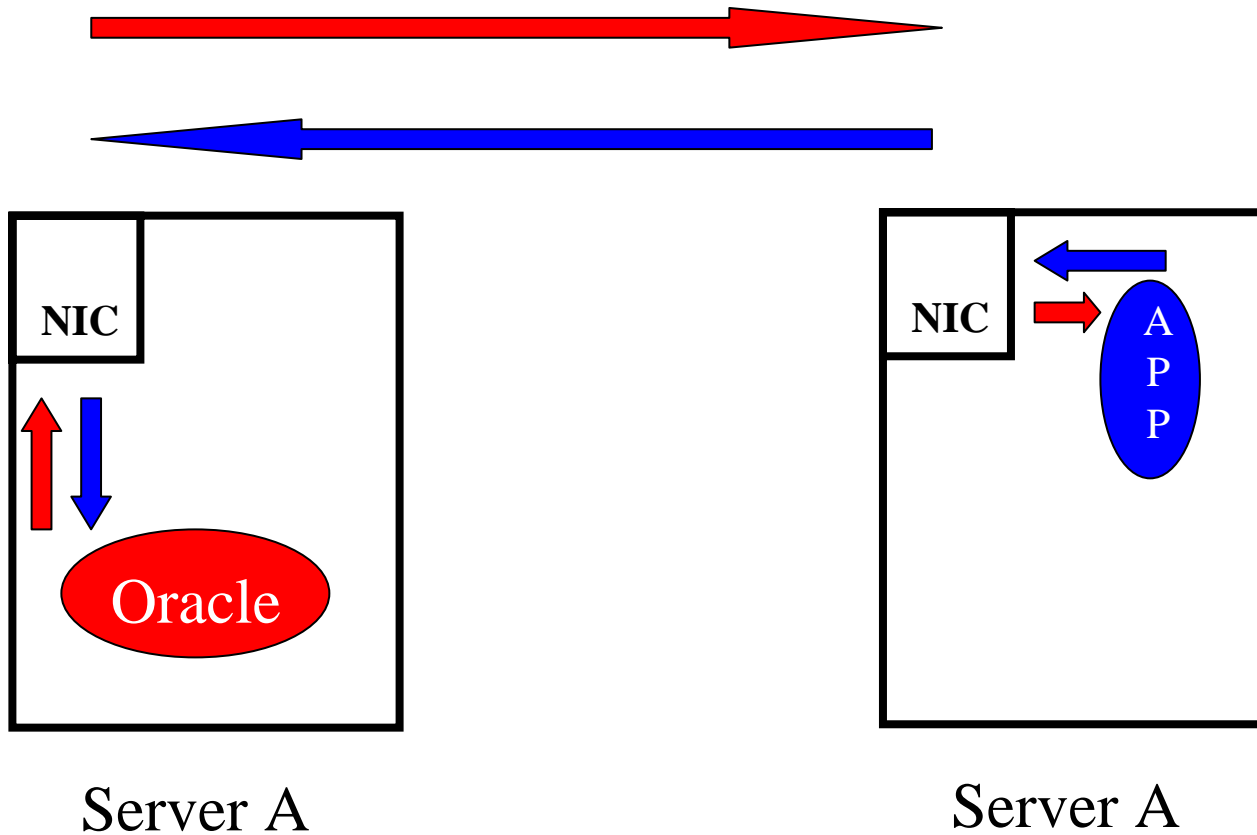
---

Response Time Component	Duration	# Calls	Avg	
SQL*Net message from client	984.01s	49.6%	95,161	0.010340s
SQL*Net more data from client	418.82s	21.1%	3,345	0.125208s
...				
Total	1,985.40s	100.0%		

- Batch program and database on same server
  - From trace data
- Payroll 'chatty' – 95,161 calls
- Very high average duration for memory based roundtrip (.01 sec)

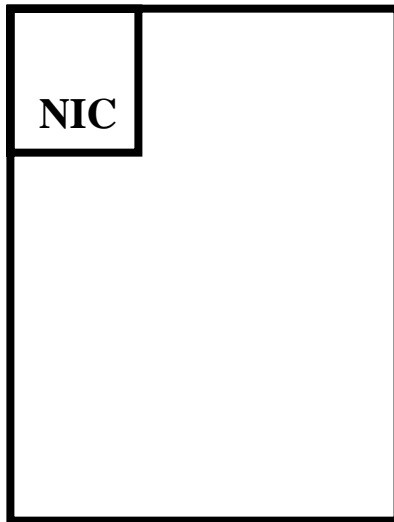
## Two server communication

---

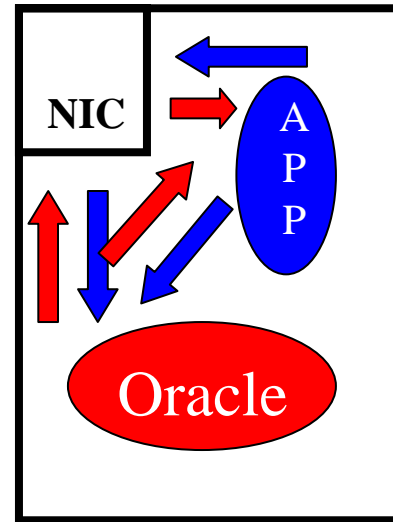


# Oracle and App on the same server – loop back

---



Server A



Server A

## High average duration: protocol = BEQ (or IPC)

---

Response Time Component	Duration	# Calls	Avg	
SQL*Net message from client	187.82s	22.1%	95,161	0.001974s
SQL*Net more data from client	80.49s	9.5%	3,345	0.024064s
...				
Total	850.89s	100.0%		

- Created new tns\_names.ora alias
  - Protocol = BEQ
- Improved from 27 to 63 assignments/min
- Found patch requiring fewer database calls (used binds)

## High average duration: Competition

---

- Your program
  - Average SQL\*Net = .03 sec
- Everyone else
  - Average SQL\*Net = .03 sec

Cursor Action	Library Misses	Action Count	Rows Processed
Parse	1	1	0
Execute	0	4	348
Fetch	0	0	0
Total	1	5	348

Cursor Action	Library Misses	Action Count	Rows Processed
Parse	348	348	0
Execute	0	348	348
Fetch	0	0	0
Total	348	696	348

Note: Useful Constants for the Oracle Performance Analyst.  
Millsap/Holt. Available at [www.hotsos.com](http://www.hotsos.com)

## High average duration: Optimize the competition

---

- They are flooding the network with unnecessary call volume
- Eventually this creates bottlenecks
- Bottlenecks create queueing
- Queueing increases response time

## Conclusion

---

- SQL\*Net message is simply a read call between database calls
- System-wide data collection drove the 'idle event' title
- If this is a top response time event you have to find the root cause
- 4 examples
  - No bind variables / redundant parsing
  - No array processing
  - Missed opportunity for protocol = BEQ or protocol = IPC
  - Optimize the competition

