

---

# Performance tuning: High CPU usage Sys mode

---

By  
Riyaj Shamsudeen



*your database maestros*

[www.pythian.com](http://www.pythian.com)

Support Centers in :  
North America | South America | Europe  
Middle East | Asia | Australia

Managed services for:  
**Oracle | SQL Server | MySQL**

# Who am I?

- 16 years using Oracle products
- Over 15 years as Oracle DBA
- Certified DBA versions 7.0,7.3,8,8i &9i
- Specializes in performance tuning, Internals and E-business suite
- Currently working for The Pythian Group [www.pythian.com](http://www.pythian.com)
- OakTable member
- Email: [rshamsud at gmail.com](mailto:rshamsud@gmail.com)
- Blog : [orainternals.wordpress.com](http://orainternals.wordpress.com)



---

# Disclaimer

These slides and materials represent the work and opinions of the author and do not constitute official positions of my current or past employer or any other organization. This material has been peer reviewed, but author assume no responsibility whatsoever for the test cases.

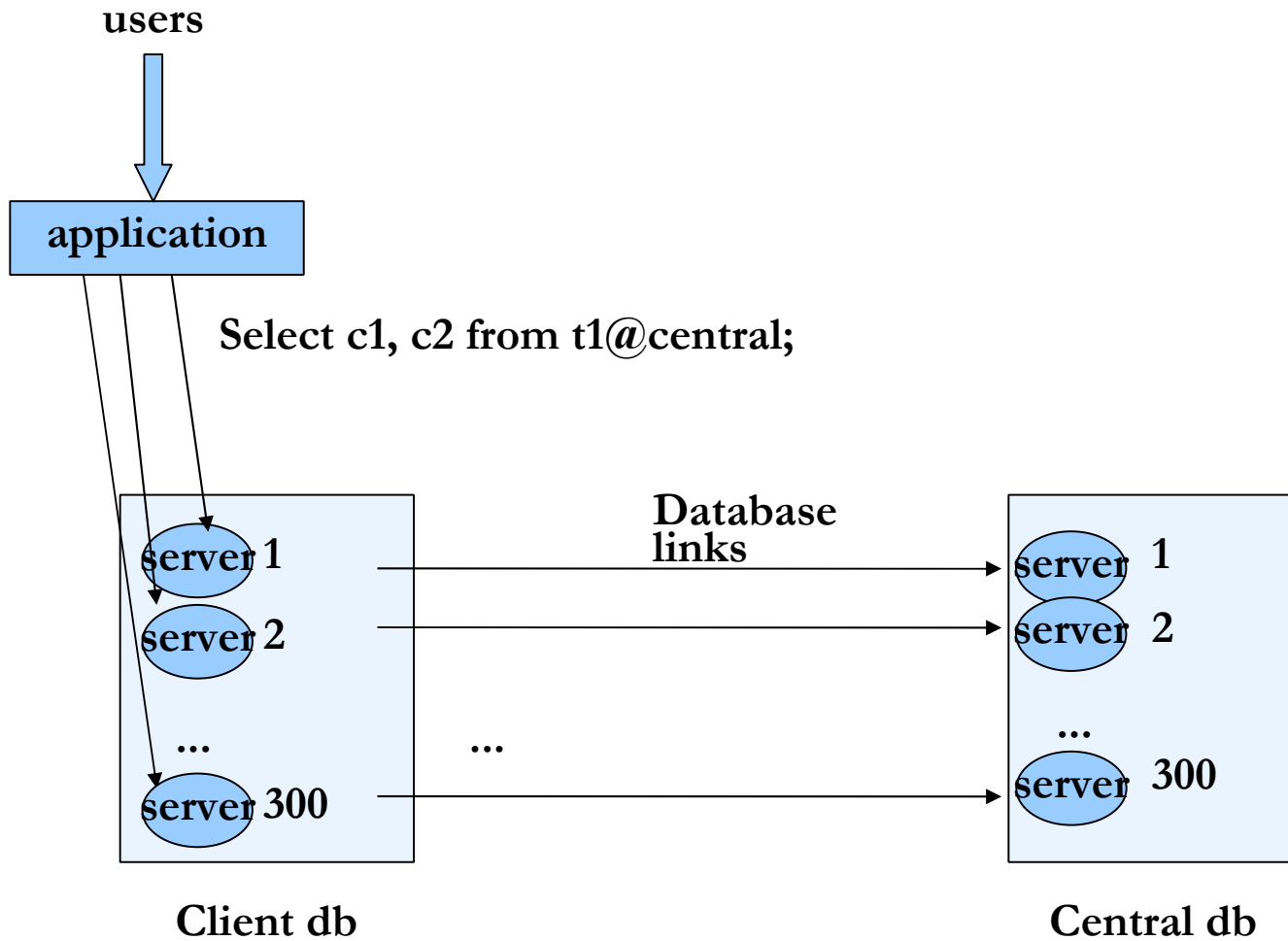
If you corrupt your databases by running my scripts, you are solely responsible for that.

This material should not should not be reproduced or used without the authors' written permission.

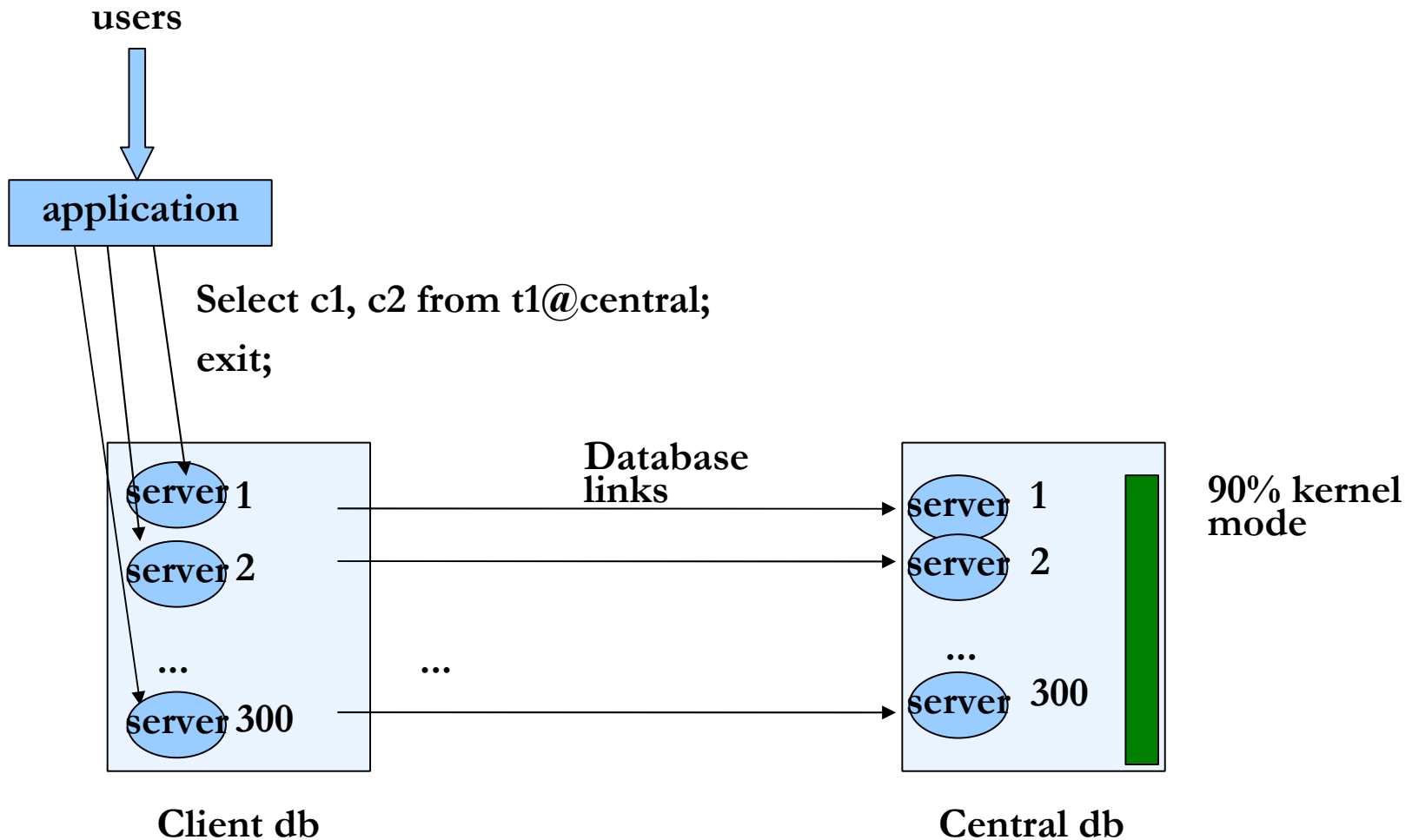
---

**Issue: High CPU usage**

# Application-Normal working



# Issue: High Kernel mode CPU usage



# Sys mode CPU usage

- Restart of an application core releases 300 database link based connections in another central database.
- This results in high kernel mode CPU usage in central database.

Tue Sep 9 17:46:34 2008	CPU	minf	mjf	xcal	intr	ithr	csw	icsw	migr	smtx	srw	syscl	usr	sys	wt	idl
Tue Sep 9 17:46:34 2008	0	561	0	9	554	237	651	219	87	491	0	4349	9	91	0	0
Tue Sep 9 17:46:34 2008	1	1197	1	34	911	0	2412	591	353	630	0	15210	30	63	0	7
Tue Sep 9 17:46:34 2008	2	58	0	9	313	0	613	106	190	105	0	3562	8	90	0	2
Tue Sep 9 17:46:34 2008	3	161	0	26	255	0	492	92	161	530	0	2914	6	92	0	2
Tue Sep 9 17:46:34 2008	4	0	0	0	86	1	2	3	1	63	0	8	0	100	0	0
Tue Sep 9 17:46:34 2008	5	283	0	34	662	0	1269	153	326	211	0	6753	13	77	0	10
Tue Sep 9 17:46:34 2008	6	434	0	43	349	0	589	54	170	1534	0	3002	7	88	0	5
Tue Sep 9 17:46:34 2008	7	142	0	17	317	81	411	105	51	61	0	2484	4	93	0	3
Tue Sep 9 17:46:34 2008	8	213	0	0	6423	6321	697	36	148	546	0	3663	6	86	0	9
Tue Sep 9 17:46:34 2008	9	291	0	17	363	0	639	41	154	1893	0	3214	9	85	0	6
Tue Sep 9 17:46:34 2008	10	149	0	8	456	0	964	194	193	77	0	5797	10	81	0	8
Tue Sep 9 17:46:34 2008	11	17	0	0	104	0	42	3	9	183	0	207	0	95	0	5
Tue Sep 9 17:46:34 2008	12	30	0	0	195	0	279	110	31	80	0	1590	3	97	0	0
Tue Sep 9 17:46:34 2008	13	288	0	9	449	0	844	117	158	127	0	4486	7	85	0	8
Tue Sep 9 17:46:34 2008	14	155	0	0	430	0	744	102	160	83	0	3875	7	80	0	13
Tue Sep 9 17:46:34 2008	15	16	0	0	237	0	359	115	31	124	0	2074	3	91	0	6

---

## Latch free

- There is also another, perceived, secondary symptom. Since all CPUs are in use, user processes will not get enough CPU cycles.
- This usually results in another symptom. In this case, latch contention on enqueues latches are another issue.
- We can't be sure that this is a secondary symptom, but kernel mode CPU usage issue must be resolved before attempting to resolve latch contention issue.

# Issue(2): Latch free

Of 655 seconds elapsed time, there were 455 seconds of waits for latch free

	Snap Id	Snap Time	Sessions	Curs/Sess
Begin Snap:	3131	09-Sep-08 17:46:17	5,030	1.9
End Snap:	3132	09-Sep-08 17:47:16	4,995	2.0
Elapsed:		0.98 (mins)		
DB Time:		10.55 (mins)		

Almost all latch free waits were for enqueues latch

Event	waits	Time (s)	(ms)	Time	Wait Class
latch free	868	455	525	71.9	Other
latch: row cache objects	103	189	1833	29.8	Concurrenc
log file sync	885	92	103	14.5	Commit
CPU time		85		13.4	
db file parallel write	3,868	10	3	1.6	System I/O

latch contention is for enqueues latches:

Latch Name	Get Requests	Pct Get Miss	Avg Slps /Miss	Wait Time (s)	Pct	
					Nowait Requests	Nowait Miss
enqueues	1,355,722	3.3	0.0	452	0	N/A

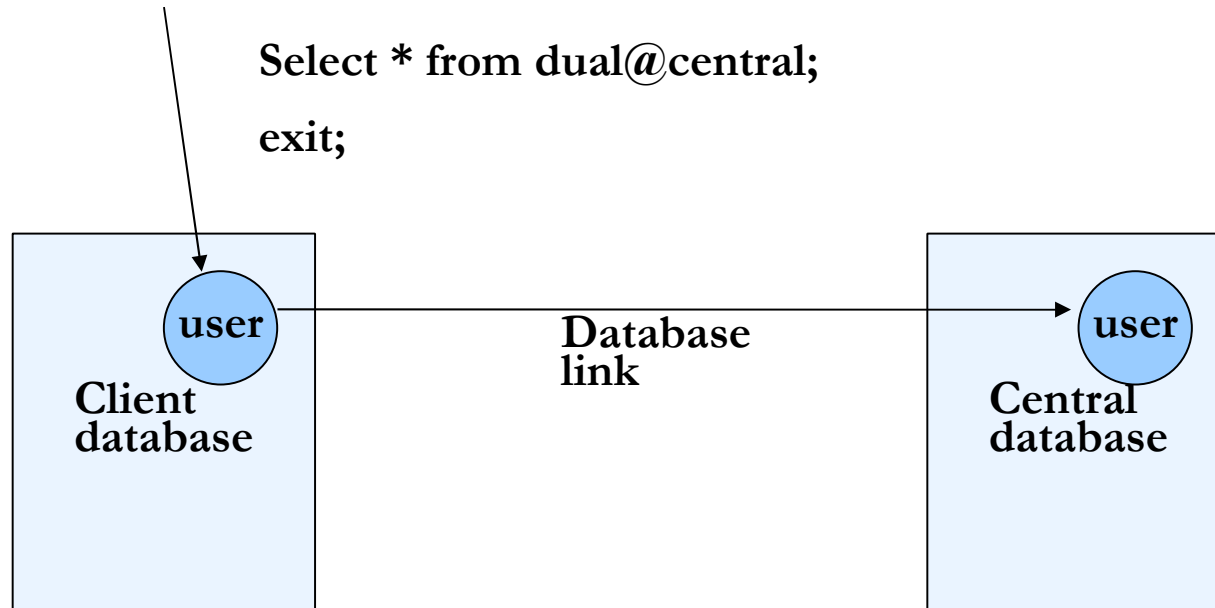
---

# Testing logoffs

- We will ignore latch contention for now.
- To test kernel mode CPU issue, we will use database link in a schema, connecting to test schema in central database.
- From client database, selecting over database link creates a new connection in central database:

```
select * from dual@central;
```

# Testing logoffs



# Truss

- Identified the connection from test schema @ client database to test schema @ central database.

```
select sid, serial#, LOGON_TIME, LAST_CALL_ET from v$session where  
logon_time > sysdate-(1/24)*(1/60)
```

SID	SERIAL#	LOGON_TIME	LAST_CALL_ET
1371	35028	12-SEP-2008 20:47:30	0
4306	51273	12-SEP-2008 20:47:29	1 <---

- Truss is an utility to trace unix system calls.
- Starting truss on that pid in central db: d flag will print offset timestamp for that system call.

```
truss -p <pid> -d -o /tmp/truss.log
```

- Logged off from content database and this should trigger a log-off from remote core database.

# Truss

- Reading truss output in central database connection, we can see ten shmdt calls are consuming time.

```
18.4630 close(10) = 0
18.4807 shmdt(0x380000000) = 0
18.5053 shmdt(0x440000000) = 0
18.5295 shmdt(0x640000000) = 0
18.5541 shmdt(0x840000000) = 0
18.5784 shmdt(0xA40000000) = 0
18.6026 shmdt(0xC40000000) = 0
18.6273 shmdt(0xE40000000) = 0
18.6512 shmdt(0x1040000000) = 0
18.6752 shmdt(0x1240000000) = 0
18.6753 shmdt(0x1440000000) = 0
```

Shmdt calls are used to detach from shared memory segments during session logoff.

$18.5295 - 18.5053 = 0.0242$

- Each call consumed approximately 0.024 seconds or 24ms.

# Shmctl calls

- There are 10 shared memory segments for this SGA. So, there are 10 shmctl calls.

```
ipcs -ma|grep 14382
```

```
m 1241514024 0x97e45100 --rw-r----- oracle orainvtr oracle orainvtr 5436
  73728 14382 1090 21:32:30 21:32:30 1:32:50
m 1241514023 0 --rw-r----- oracle orainvtr oracle orainvtr 5436
  8153726976 14382 1090 21:32:30 21:32:30 1:32:42
m 1241514022 0 --rw-r----- oracle orainvtr oracle orainvtr 5436
  8187281408 14382 1090 21:32:30 21:32:30 1:32:34
m 1241514021 0 --rw-r----- oracle orainvtr oracle orainvtr 5436
  8153726976 14382 1090 21:32:30 21:32:30 1:32:26
m 1241514020 0 --rw-r----- oracle orainvtr oracle orainvtr 5436
  8153726976 14382 1090 21:32:30 21:32:30 1:32:18
m 1241514019 0 --rw-r----- oracle orainvtr oracle orainvtr 5436
  8153726976 14382 1090 21:32:30 21:32:30 1:32:11
m 1241514018 0 --rw-r----- oracle orainvtr oracle orainvtr 5436
  8153726976 14382 1090 21:32:30 21:32:30 1:32:03
m 1241514017 0 --rw-r----- oracle orainvtr oracle orainvtr 5436
  8153726976 14382 1090 21:32:30 21:32:30 1:31:56
m 1241514016 0 --rw-r----- oracle orainvtr oracle orainvtr 5436
  8153726976 14382 1090 21:32:30 21:32:30 1:31:49
m 889192479 0 --rw-r----- oracle orainvtr oracle orainvtr 5436
  2986344448 14382 1090 21:32:30 21:32:30 1:31:47
```

8GB

---

## Fun with numbers

- Each session consumes approximately 0.24 seconds. Shmctl is a system call and so cpu usage will be in kernel mode.
- 300 connections would consume 72 seconds.
- But, as concurrency goes up, these calls will slow down and use more mutex spins, which in turn will increase kernel mode cpu usage.
- At best case, with 12 concurrent processes, this would last for 6 seconds or so.
- This is matching with our observation of 6 seconds high kernel mode cpu usage.

---

## Reducing shmdt calls

- Number of shmdt calls can be reduced by having one shared memory segment.
- Database engine tries to create biggest segment possible at initial startup and slowly reduces segment size, until segments can be created successfully.
- In this case, SHMMAX kernel parameter value is limiting and that's why RDBMS is creating ten segments, instead of one.
- So, to reduce number of shared memory segments, we need to increase or set SHMMAX to maximum value. There is no downside to that.
- This will likely require a server reboot.
- This change will also necessitate database restart too.

---

# dtrace

- While we are almost certain that shmdt calls are causing this issue, we need to perform a verification analysis too.
- Dtrace can be used to see what calls are executed by peeking at CPUs.
- It is a possibility that dtrace might give a different view of the problem, but still it is prudent to perform this analysis too.
- Following dtrace command to be executed right before app core is shutdown and should be ctrl+C after CPU usage subsides.

```
dtrace -n 'syscall:::entry { @Calls[probefunc] = count(); }'
```

- For comparison, we should also execute above command for same number of seconds when it is considered normal.

---

## Dtrace – top sys call

- We also need to use `top_sys_calls.sh` to understand top sys calls per second basis using `top_sys_calls.sh` script.
- Also `dtrace` is considered as a safe option in Solaris platform.

# After SHMMAX change

- After SHMMAX change, we expected SGA to be fit in one segment.
- Surprise! Surprise! Surprise!
- Started to truss on database startup to see why the instance is creating multiple shared memory segments.

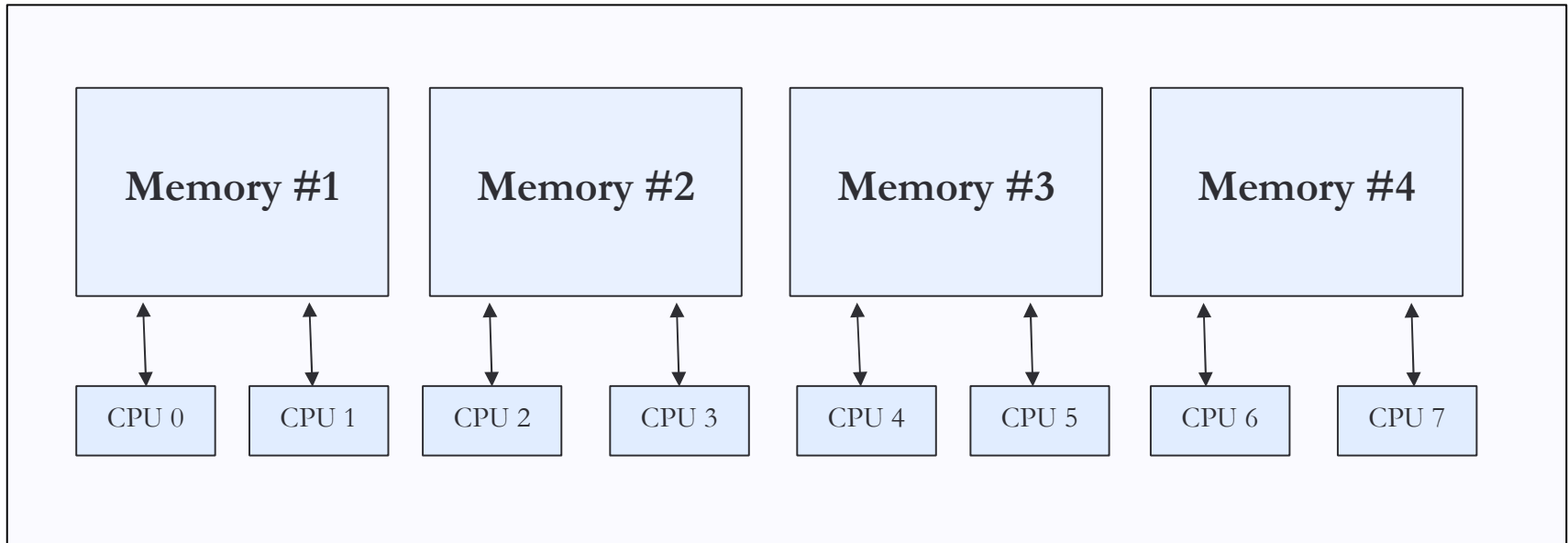
```
17252: 4.5957 munmap(0xFFFFFD7FFDAE0000, 32768) = 0
17252: 4.5958 lgrp_version(1, ) = 1
17252: 4.5958 _lgrpsys(1, 0, ) = 42
17252: 4.5958 _lgrpsys(3, 0x00000000, 0x00000000) = 19108
17252: 4.5959 _lgrpsys(3, 0x00004AA4, 0x06399D60) = 19108
17252: 4.5959 _lgrpsys(1, 0, ) = 42
17252: 4.5960 pset_bind(PS_QUERY, P_LWPID, 4294967295, 0xFFFFFD7FFDFB11C) = 0
17252: 4.5960 pset_info(PS_MYID, 0x00000000, 0xFFFFFD7FFDFB0D4, 0x00000000) = 0
17252: 4.5961 pset_info(PS_MYID, 0x00000000, 0xFFFFFD7FFDFB0D4, 0x061AA2B0) = 0
```

# pset\_bind and \_lgrpsys

- pset\_bind and \_lgrpsys gave a clue that this might be related to NUMA (Non-uniform memory access) architecture issues.

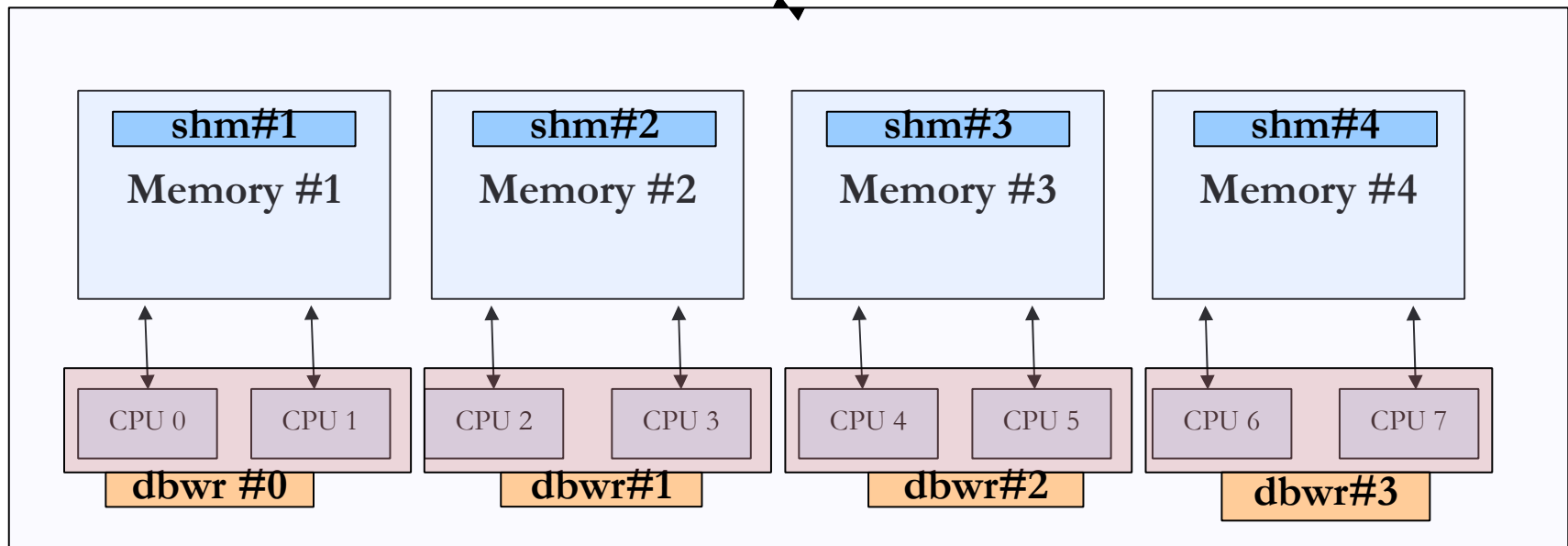
```
17252: 4.5957 munmap(0xFFFFFD7FFDAE0000, 32768) = 0
17252: 4.5958 lgrp_version(1, ) = 1
17252: 4.5958 _lgrpsys(1, 0, ) = 42
17252: 4.5958 _lgrpsys(3, 0x00000000, 0x00000000) = 19108
17252: 4.5959 _lgrpsys(3, 0x00004AA4, 0x06399D60) = 19108
17252: 4.5959 _lgrpsys(1, 0, ) = 42
17252: 4.5960 pset_bind(PS_QUERY, P_LWPID, 4294967295, 0xFFFFFD7FFDFB11C) = 0
17252: 4.5960 pset_info(PS_MYID, 0x00000000, 0xFFFFFD7FFDFB0D4, 0x00000000) = 0
17252: 4.5961 pset_info(PS_MYID, 0x00000000, 0xFFFFFD7FFDFB0D4, 0x061AA2B0) = 0
```

# NUMA architecture (High level overview only)



- For cpu0 & cpu1, memory #1 is local. Other memory areas are remote for cpu0 & cpu1.
- Access to local memory is very fast compared to remote memory access.

# NUMA architecture (High level overview only)



- To make use of NUMA technology, Oracle spreads SGA across all memory areas.
- Then Binds DBWR to a CPU set. That DBWR handles all writes from that shared memory segment. LGWR is also bound to a processor set.

---

## Back to our issue

- In Solaris, NUMA technology is implemented as locality groups.
- `_lgrpsys` and `pset_bind` calls are to get current locality group information and bind processes to a processor set.
- Now, we can understand why SGA was split into multiple segments.
- But, Do we really have that many locality groups in this server?

# Locality groups

- With the help of an eminent SA, we were able to download locality group aware tools from openSolaris.org and execute that in that platform.

```
:~#/usr/local/bin/lgrpinfo
```

```
lgroup 0 (root):
```

```
Children: 10 12 14 15 17 19 21 23
```

```
CPUs: 0-15
```

```
Memory: installed 65024 Mb, allocated 2548 Mb, free 62476 Mb
```

```
Lgroup resources: 1-8 (CPU); 1-8 (memory)
```

```
Latency: 146
```

```
lgroup 1 (leaf):
```

```
Children: none, Parent: 9
```

```
CPUs: 0 1
```

```
Memory: installed 7680 Mb, allocated 1964 Mb, free 5716 Mb
```

```
Lgroup resources: 1 (CPU); 1 (memory)
```

```
Load: 0.105
```

```
Latency: 51
```

**There were many locality groups defined and seven of them were leaf node locality groups.**

**Locality groups are defined as an hierarchical tree.**

```
...
```

# Latency

---

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	
0	146	146	113	113	113	113	113	113	146	146	146	146	146	146	146	113	146	113	146	146	146	146	146	146	146	
1	146	51	81	81	113	113	113	113	146	81	113	113	146	113	146	146	113	146	113	146	146	146	146	146	146	146
2	113	81	51	113	81	113	113	81	113	113	113	81	113	113	113	113	113	113	113	113	113	113	113	113	113	113
3	113	81	113	51	113	81	81	113	113	113	113	113	113	81	113	113	113	113	113	113	113	113	113	113	113	113
4	113	113	81	113	51	81	81	113	113	113	113	113	113	113	113	113	81	113	113	113	113	113	113	113	113	113
5	113	113	113	81	81	51	113	81	113	113	113	113	113	113	113	113	113	113	81	113	113	113	113	113	113	113
6	113	113	113	81	81	113	51	113	81	113	113	113	113	113	113	113	113	113	113	113	81	113	113	113	113	113
7	113	113	81	113	113	81	113	51	81	113	113	113	113	113	113	113	113	113	113	113	113	113	113	81	113	113
8	146	146	113	113	113	113	81	81	51	146	146	146	146	146	146	146	113	146	113	146	113	146	113	113	81	113
9	146	81	113	113	113	113	113	113	146	81	113	113	146	113	146	146	113	146	113	146	146	146	146	146	146	146
10	146	113	113	113	113	113	113	113	146	113	113	113	146	113	146	146	113	146	113	146	146	146	146	146	146	146
11	146	113	81	113	113	113	113	113	146	113	113	81	146	113	146	146	113	146	113	146	146	146	146	146	146	146
12	146	146	113	113	113	113	113	113	146	146	146	146	113	146	146	146	113	146	113	146	146	146	146	146	146	146
13	146	113	113	81	113	113	113	113	146	113	113	113	146	81	146	146	113	146	113	146	146	146	146	146	146	146
14	146	146	113	113	113	113	113	113	146	146	146	146	146	146	113	146	113	146	113	146	146	146	146	146	146	146
15	146	146	113	113	113	113	113	113	146	146	146	146	146	146	146	113	113	146	113	146	146	146	146	146	146	146
...																										

---

---

# Summary

- Indeed 10 shared memory segments were created, one for a locality groups.
- We can disable NUMA or reduce number of NUMA nodes.
  - \*.\_enable\_NUMA\_optimization=FALSE
  - \*.\_db\_block\_numa = <smaller number>
- Of course, these are underscore parameters and so need Oracle support.
- Note 399261.1 describes that.
- It looks like, there is one shared memory segment per locality group, one encompassing all locality groups and one small bootstrap segment.

---

# References

- Oracle support site. [Metalink.oracle.com](http://Metalink.oracle.com). Various documents
- Internal's guru Steve Adam's website  
[www.ixora.com.au](http://www.ixora.com.au)
- Jonathan Lewis' website  
[www.jlcomp.daemon.co.uk](http://www.jlcomp.daemon.co.uk)
- Julian Dyke's website  
[www.julian-dyke.com](http://www.julian-dyke.com)
- 'Oracle8i Internal Services for Waits, Latches, Locks, and Memory'  
by Steve Adams
- Tom Kyte's website  
[Asktom.oracle.com](http://Asktom.oracle.com)
- Blog: <http://orainternals.wordpress.com>