OakTable World

DOAG

# Minimizing the Concurrency Footprint of Transactions

Mark W. Farnham

Rightsizing, Inc.

RIGHTSIZING INC

Oracle Database Technology and Business Consulting

OakTable®

# Mark W. Farnham

President, Rightsizing, Inc.

Co-founding Director, OAUG

Charter member, current director IOUG

Secretary, OAUG DBSIG

Member, MOSES, Oracle VLDB

Co-founder, APPSPERF

Member, Oaktable Network

Oracle Ace



RIGHTSIZING INC

IOUG
independent oracle users group
For the Complete Technology & Database Professional

DBSIG

# Mark W. Farnham

Proud to be included on this agenda

US Senate primary candidate in 1992 and 2014

Soundly crushed by a statesman in 1992

Denied a spot on the TV debate in 2014

This is NOT a political talk, needed to explain OTW come-on tweet

So, let's talk some technology!



RIGHTSIZING INC


For the Complete Technology & Database Professional

# Complement your Oaktable World and DOAG experience


- So many good sessions at DOAG you can't see them all?
- It's a good bet some great sessions will be repeated in Las Vegas!

# Agenda

- Told you a bit about me, made pitch for IOUG
- Frame what the heck I'm talking about
- Tell you to instrument your code
- Study ways to submit a trivial transaction
  - In tedious *but important* detail
    - Differences in how values get from client to server
      - SQL*Plus substitution variables
      - defined bind variables
  - SQL*Plus convenient, not a transaction engine

# Agenda (continued)

- Considerations from trivial case study
  - Model procedures for use by transaction engines
  - Lower complexity of bits that must be repeatedly parsed
  - Ship a small call, not the whole code
  - Start the transaction when you have everything you need to finish it already on the server
- Briefer study of a more complex transaction
- Establish an order to minimize deadlocks
- On beyond zebra

# What the heck I'm talking about

- Concurrency Footprint
  - Duration of possibly blocking something else
  - Avoid unneeded increases of the footprint
  - Design with room for footprint reduction
    - Lazy initial is okay if you can improve later
    - Minimize initial complexity
    - Minimize extra work unless needed
    - But the plan is in place
- You care if transaction rates rise

# What the heck I'm talking about

- Small transactions
  - Probably a simple plan and one row is quick
    - Note the difference between:
      - Processing a large set one row at a time (not smart)
      - Processing a single row when that is the whole set
    - If not quick, probably need to make it quick
  - Line turn-around is a significant fraction of the time
  - Logical Units of work
    - Usually one or a few rows
    - Multiple tables before commit
- Same Order for different LUWs is important

# Instrument Your Code

- All of it
  - https:/blogs.oracle.com/archbeat/entry/top_ten_2_minutes_tech
  - Cary Millsap – "Trace Your Code – All of it."
- PL/SQL is excellent for instrumentation
  - www.method-r.com
  - Performance Instrumentation for PL/SQL: When, Why, How (PPT)
  - Karen Morton, December 2008
  - Native calls or ILO – instrument it all

# Instrument Your Code (continued)

- Logic is logic
  - If you can assemble a query on the client
  - Then you can also assemble it in PL/SQL
  - Far easier to instrument in PL/SQL wrapper
  - Choices can still be computed on client
    - Avoid burning DB licensed CPU
    - Facilitate horizontal scaling on cheaper client CPU
    - Likely possible to enumerate *allowed transactions*
  - Shipping tokens to reflect choices
    - More compact
    - At least very difficult to inject with code strings
  - Don't start the transaction until everything needed is at the server! Seriously. Even if "super network"

# Ways to submit a trivial transaction

- Send in code from client and then commit it
- Send in a PL/SQL block from client
- Execute a stored procedure
  - with SQL*Plus substitution variables
  - with values bound to defined variables
- Value binding required for all but literals
- Some parsing always required from SQL*Plus to interpret values with variables
- Which way translates easier to a program?

# Send in code from client, then commit

```
update inventory
   set onhand = onhand - &purchase_qty
   where skucs = &skuc_s;
commit;
```

# Send in code from client, then commit

```
WAIT #3: nam='SQL*Net message from client' ela= 7858361 driver id=1111838976 #bytes=1 p3=0 obj#=116
tim=2049435525067
CLOSE #3:c=0,e=99,dep=0,type=0,tim=2049435525303
=====================
PARSING IN CURSOR #5 len=58 dep=0 uid=91 oct=6 lid=91 tim=2049435526367 hv=1676997574
ad='7ffbe300910' sqlid='1h56wm1jz9wy6'
update inventory set onhand = onhand - 100
where skucs = 1
END OF STMT
PARSE #5:c=0,e=1009,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=1,plh=2762941418,tim=2049435526366
EXEC #5:c=0,e=146,p=0,cr=1,cu=3,mis=0,r=1,dep=0,og=1,plh=2762941418,tim=2049435526606
STAT #5 id=1 cnt=0 pid=0 pos=1 obj=0 op='UPDATE  INVENTORY (cr=1 pr=0 pw=0 time=0 us)'
STAT #5 id=2 cnt=1 pid=1 pos=1 obj=178147 op='INDEX UNIQUE SCAN INVENTORY_PK (cr=1 pr=0 pw=0 time=0
us cost=0 size=6 card=1)'
WAIT #5: nam='SQL*Net message to client' ela= 4 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2049435526718
WAIT #5: nam='SQL*Net message from client' ela= 2388 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2049435529133
CLOSE #5:c=0,e=16,dep=0,type=0,tim=2049435529282
=====================
PARSING IN CURSOR #4 len=6 dep=0 uid=91 oct=44 lid=91 tim=2049435529331 hv=3480936638 ad='0'
sqlid='23wm3kz7rps5y'
commit
END OF STMT
```

# Send in a PL/SQL block from client

```
variable purchase_qty number
variable skucs number
begin
 :purchase_qty := 100;
 :skucs := 1;
 update inventory set onhand = onhand - :purchase_qty
   where skucs = :skucs;
 commit;
 end;
 /
```

# Send in a PL/SQL block from client

```
PARSING IN CURSOR #2 len=128 dep=0 uid=91 oct=47 lid=91 tim=2053378728848 hv=4238991900 ad='7ffb5b6ed40'
sqlid='7t8134myamshw'
begin
:purchase_qty := 100;
:skucs := 1;
update inventory set onhand = onhand - :purchase_qty
where skucs = :skucs;
commit;
end;
END OF STMT
PARSE #2:c=0,e=148,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=0,tim=2053378728847
BINDS #2:
 Bind#0
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=48 off=0
  kxsbbbfp=1c2f2430  bln=22  avl=00  flg=05
 Bind#1
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=24
  kxsbbbfp=1c2f2448  bln=22  avl=00  flg=01
=====================
```

# Send in a PL/SQL block from client

```
PARSING IN CURSOR #4 len=60 dep=1 uid=91 oct=6 lid=91 tim=2053378729409 hv=3209938921 ad='7ffbee1d3e0'
sqlid='2zy5fhkzp7jz9'
UPDATE INVENTORY SET ONHAND = ONHAND - :B1 WHERE SKUCS = :B2
END OF STMT
PARSE #4:c=0,e=257,p=0,cr=0,cu=0,mis=1,r=0,dep=1,og=1,plh=0,tim=2053378729408
BINDS #4:
 Bind#0
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=202001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1c2f2430  bln=22  avl=02  flg=09
  value=100
 Bind#1
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=202001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1c2f2448  bln=22  avl=02  flg=09
  value=1
EXEC #4:c=0,e=1225,p=0,cr=1,cu=3,mis=1,r=1,dep=1,og=1,plh=2762941418,tim=2053378730701
STAT #4 id=1 cnt=0 pid=0 pos=1 obj=0 op='UPDATE  INVENTORY (cr=1 pr=0 pw=0 time=0 us)'
STAT #4 id=2 cnt=1 pid=1 pos=1 obj=178147 op='INDEX UNIQUE SCAN INVENTORY_PK (cr=1 pr=0 pw=0 time=0 us
cost=0 size=6 card=1)'
CLOSE #4:c=0,e=2,dep=1,type=3,tim=2053378730824
=====================
PARSING IN CURSOR #3 len=6 dep=1 uid=91 oct=44 lid=91 tim=2053378730860 hv=255718823 ad='0'
sqlid='8ggw94h7mvxd7'
COMMIT
END OF STMT
```

- Notice no SQL*net wait pair; repeats don't miss

# Don’t stop there! We can do *much* better

- Still had to ship all the code; we can do much better with a packaged procedure stored in the database
- Can instrument without adding code to ship
- Can be kept in shared pool
  - Don’t *have to* – Oracle handles it
  - *Avoid loading during user experience if streaky*
  - *Pre-load big things at slack time*
  - *Can manage with dummy heartbeat calls*

# Barebones Stored procedure

- Still no instrumentation (small enough to see)
- Exec traces with substitution variables and defined variables follow

```
procedure minus_inventory_p1(p_skucs number, p_purchase_qty number) is
   --
   begin
      update inventory i set i.onhand = i.onhand - p_purchase_qty
         where i.skucs = p_skucs;
      commit;
   --
end minus_inventory_p1;
```

# Barebones Stored procedure

```
procedure minus_inventory_p1(p_skucs number, p_purchase_qty number) is
   --
   begin
      update inventory i set i.onhand = i.onhand - p_purchase_qty
         where i.skucs = p_skucs;
      commit;
   --
end minus_inventory_p1;
```

- Three submission examples
  - exec with substitution variables (literals)
  - one pl/sql block with defined (bind) variables
  - two pl/sql blocks
    - set values
    - execute block

# Barebones Stored Procedure -literals

exec sell_pkg.minus_inventory_p1(&skucs,&purchase_qty);

```
PARSING IN CURSOR #1 len=46 dep=0 uid=91 oct=47 lid=91 tim=2359132560458 hv=1859270037 ad='7ffbe698cd8'
sqlid='6585ph9rd4dcp'
BEGIN sell_pkg.minus_inventory_p1(1,1); END;
END OF STMT
PARSE #1:c=0,e=1522,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=1,plh=0,tim=2359132560457
=====================
PARSING IN CURSOR #4 len=69 dep=1 uid=91 oct=6 lid=91 tim=2359132560802 hv=391566703 ad='7ffbe533c20'
sqlid='52cqs6wbpdpbg'
UPDATE INVENTORY I SET I.ONHAND = I.ONHAND - :B2 WHERE I.SKUCS = :B1
END OF STMT
PARSE #4:c=0,e=157,p=0,cr=0,cu=0,mis=1,r=0,dep=1,og=1,plh=0,tim=2359132560802
BINDS #4:
 Bind#0
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1bf43a68  bln=22  avl=02  flg=09
  value=1
 Bind#1
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1bf43a68  bln=22  avl=02  flg=09
  value=1
EXEC #4:c=0,e=933,p=0,cr=1,cu=3,mis=1,r=1,dep=1,og=1,plh=2762941418,tim=2359132561791
STAT #4 id=1 cnt=0 pid=0 pos=1 obj=0 op='UPDATE  INVENTORY (cr=1 pr=0 pw=0 time=0 us)'
STAT #4 id=2 cnt=1 pid=1 pos=1 obj=178147 op='INDEX UNIQUE SCAN INVENTORY_PK (cr=1 pr=0 pw=0 time=0 us
cost=0 size=7 card=1)'
CLOSE #4:c=0,e=2,dep=1,type=3,tim=2359132561886
=====================
PARSING IN CURSOR #2 len=6 dep=1 uid=91 oct=44 lid=91 tim=2359132561925 hv=255718823 ad='0'
sqlid='8ggw94h7mvxd7'
COMMIT
```

# Barebones Stored Procedure – 1 block

```
variable purchase_qty number
variable skucs number

begin
:purchase_qty := &qty;
:skucs := &skucs;
sell_pkg.minus_inventory_p1(:skucs,:purchase_qty);
end;
/
```

# Barebones Stored Procedure – 1 block

```
PARSING IN CURSOR #4 len=94 dep=0 uid=91 oct=47 lid=91 tim=2370568927053 hv=1280339793
ad='7ffbe776478' sqlid='5ctqpcj650vuj'
begin
:purchase_qty := 3;
:skucs := 1;
sell_pkg.minus_inventory_p1(:skucs,:purchase_qty);
end;
END OF STMT
PARSE #4:c=0,e=544,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=1,plh=0,tim=2370568927053
BINDS #4:
 Bind#0
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=48 off=0
  kxsbbbfp=0d2756e8  bln=22  avl=00  flg=05
 Bind#1
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=24
  kxsbbbfp=0d275700  bln=22  avl=00  flg=01
=====================
PARSING IN CURSOR #7 len=69 dep=1 uid=91 oct=6 lid=91 tim=2370568928461 hv=391566703 ad='7ffbe533c20'
sqlid='52cqs6wbpdpbg'
UPDATE INVENTORY I SET I.ONHAND = I.ONHAND - :B2 WHERE I.SKUCS = :B1
END OF STMT
PARSE #7:c=0,e=27,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=2762941418,tim=2370568928460
BINDS #7:
 Bind#0
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1da23a68  bln=22  avl=02  flg=09
  value=3
 Bind#1
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1da23a98  bln=22  avl=02  flg=09
  value=1
EXEC #7:c=0,e=290,p=0,cr=1,cu=3,mis=0,r=1,dep=1,og=1,plh=2762941418,tim=2370568928839
STAT #7 id=1 cnt=0 pid=0 pos=1 obj=0 op='UPDATE  INVENTORY (cr=1 pr=0 pw=0 time=0 us)'
STAT #7 id=2 cnt=1 pid=1 pos=1 obj=178147 op='INDEX UNIQUE SCAN INVENTORY_PK (cr=1 pr=0 pw=0 time=0
us cost=0 size=7 card=1)'
CLOSE #7:c=0,e=1,dep=1,type=3,tim=2370568928956
=====================
PARSING IN CURSOR #5 len=6 dep=1 uid=91 oct=44 lid=91 tim=2370568928992 hv=255718823 ad='0'
sqlid='8ggw94h7mvxd7'
COMMIT
```

# Barebones Stored Procedure –2 blocks

```
begin
:purchase_qty := 42;
:skucs := 1;
end;
/

exec sell_pkg.minus_inventory_p1(:skucs,:purchase_qty);

begin
:purchase_qty := 54;
:skucs := 1;
end;
/

exec sell_pkg.minus_inventory_p1(:skucs,:purchase_qty);
```

# Barebones Stored Procedure –2 blocks

```
PARSING IN CURSOR #3 len=44 dep=0 uid=91 oct=47 lid=91 tim=2579940374686 hv=118800706
ad='7ffbe21f320' sqlid='6w6tg383j9ha2'
begin
:purchase_qty := 42;
:skucs := 1;
end;
END OF STMT
PARSE #3:c=0,e=487,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=1,plh=0,tim=2579940374685
BINDS #3:
 Bind#0
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=48 off=0
  kxsbbbfp=1ef905e0  bln=22  avl=00  flg=05
 Bind#1
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=24
  kxsbbbfp=1ef905f8  bln=22  avl=00  flg=01
WAIT #3: nam='SQL*Net message to client' ela= 2 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2579940375714
EXEC #3:c=0,e=979,p=0,cr=0,cu=0,mis=1,r=1,dep=0,og=1,plh=0,tim=2579940375735

*** 2015-06-12 15:21:55.310
WAIT #3: nam='SQL*Net message from client' ela= 20895793 driver id=1111838976 #bytes=1 p3=0 obj#=-
tim=2579961271586
```

CLOSE #3:c=0,e=52,dep=0,type=0,tim=2579961271817

But notice these waits are before the transaction starts!

# Barebones Stored Procedure –2 blocks

```
PARSING IN CURSOR #2 len=63 dep=0 uid=91 oct=47 lid=91 tim=2579961272175 hv=3736181483
ad='7ffbeea9bd0' sqlid='fjx4svggb36rb'
BEGIN sell_pkg.minus_inventory_p1(:skucs,:purchase_qty); END;
END OF STMT
PARSE #2:c=0,e=303,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=0,tim=2579961272174
BINDS #2:
 Bind#0
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=48 off=0
  kxsbbbfp=1ef905e0  bln=22  avl=02  flg=05
  value=1
 Bind#1
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=24
  kxsbbbfp=1ef905f8  bln=22  avl=02  flg=01
  value=42
=====================
PARSING IN CURSOR #3 len=69 dep=1 uid=91 oct=6 lid=91 tim=2579961272597 hv=391566703 ad='7ffbe533c20'
sqlid='52cqs6wbpdpbg'
UPDATE INVENTORY I SET I.ONHAND = I.ONHAND - :B2 WHERE I.SKUCS = :B1
END OF STMT
PARSE #3:c=0,e=26,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=2762941418,tim=2579961272596
BINDS #3:
 Bind#0
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1da63a98  bln=22  avl=02  flg=09
  value=42
 Bind#1
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1da63a68  bln=22  avl=02  flg=09
  value=1
EXEC #3:c=0,e=320,p=0,cr=1,cu=3,mis=0,r=1,dep=1,og=1,plh=2762941418,tim=2579961273002
STAT #3 id=1 cnt=0 pid=0 pos=1 obj=0 op='UPDATE  INVENTORY (cr=1 pr=0 pw=0 time=0 us)'
STAT #3 id=2 cnt=1 pid=1 pos=1 obj=178147 op='INDEX UNIQUE SCAN INVENTORY_PK (cr=1 pr=0 pw=0 time=0
us cost=0 size=7 card=1)'
CLOSE #3:c=0,e=2,dep=1,type=3,tim=2579961273137
====================
PARSING IN CURSOR #3 len=6 dep=1 uid=91 oct=44 lid=91 tim=2579961273180 hv=255718823 ad='0'
sqlid='8ggw94h7mvxd7'
COMMIT
```

# Barebones Stored Procedure –2 blocks

```
PARSING IN CURSOR #3 len=44 dep=0 uid=91 oct=47 lid=91 tim=2580229657325 hv=1563600720
ad='7ffb5f62280' sqlid='8dbnw9jfm59uh'
begin
:purchase_qty := 54;
:skucs := 1;
end;
END OF STMT
PARSE #3:c=0,e=542,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=1,plh=0,tim=2580229657324
BINDS #3:
 Bind#0
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=48 off=0
  kxsbbbfp=1ef909b0  bln=22  avl=00  flg=05
 Bind#1
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=24
  kxsbbbfp=1ef909c8  bln=22  avl=00  flg=01
WAIT #3: nam='SQL*Net message to client' ela= 4 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2580229658836
EXEC #3:c=0,e=1421,p=0,cr=0,cu=0,mis=1,r=1,dep=0,og=1,plh=0,tim=2580229658870

*** 2015-06-12 15:26:36.070
WAIT #3: nam='SQL*Net message from client' ela= 12367948 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2580242026902
```

# Barebones Stored Procedure –2 blocks

```
PARSING IN CURSOR #2 len=63 dep=0 uid=91 oct=47 lid=91 tim=2580242027239 hv=3736181483
ad='7ffbeea9bd0' sqlid='fjx4svggb36rb'
BEGIN sell_pkg.minus_inventory_p1(:skucs,:purchase_qty); END;
END OF STMT
PARSE #2:c=0,e=68,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=0,tim=2580242027237
BINDS #2:
 Bind#0
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=48 off=0
  kxsbbbfp=1ef909b0  bln=22  avl=02  flg=05
  value=1
 Bind#1
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=24
  kxsbbbfp=1ef909c8  bln=22  avl=02  flg=01
  value=54
=====================
PARSING IN CURSOR #3 len=69 dep=1 uid=91 oct=6 lid=91 tim=2580242027582 hv=391566703 ad='7ffbe533c20'
sqlid='52cqs6wbpdpbg'
UPDATE INVENTORY I SET I.ONHAND = I.ONHAND - :B2 WHERE I.SKUCS = :B1
END OF STMT
BINDS #3:
 Bind#0
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1da63a98  bln=22  avl=02  flg=09
  value=54
 Bind#1
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1da63a68  bln=22  avl=02  flg=09
  value=1
EXEC #3:c=0,e=396,p=0,cr=1,cu=3,mis=0,r=1,dep=1,og=1,plh=2762941418,tim=2580242027947
CLOSE #3:c=0,e=2,dep=1,type=3,tim=2580242028018
XCTEND rlbk=0, rd_only=0, tim=2580242028058
=====================
PARSING IN CURSOR #3 len=6 dep=1 uid=91 oct=44 lid=91 tim=2580242028171 hv=255718823 ad='0'
sqlid='8ggw94h7mvxd7'
COMMIT
```

# Barebones Stored Procedure – review

- Whew!
- Exec with subst. vars. reparses for any change
  - cursor_sharing NO help
  - not really a cursor, does require library latch
- Single block reparses even with bind vars
- Two blocks
  - Reparses the very simple values block
  - Just binds and executes the “call”
  - (Did I remember to note the smaller parse time?)
  - Better models OCIBind... calls from programs

# Multi-statement Logical Unit of Work

- Sell and ship
  - some amount
  - of something
  - top someone
- Still pretty simple
- Lots of opportunity to speed up comparing
  - client shipped code
  - stored procedure

# LUW sqlplus bind

```
PARSING IN CURSOR #3 len=58 dep=0 uid=91 oct=47 lid=91 tim=189900323069 hv=
356505772 ad='7ffbf2d8cb0' sqlid='953w0nsamzq5c'
begin
:purchase_qty := 54;
:skucs := 2;
:custid := 1;
end;
END OF STMT
PARSE #3:c=0,e=560,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=1,plh=0,tim=189900323068
BINDS #3:
 Bind#0
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=72 off=0
  kxsbbbfp=1cecffb8  bln=22  avl=00  flg=05
 Bind#1
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=24
  kxsbbbfp=1cecffd0  bln=22  avl=00  flg=01
 Bind#2
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=48
  kxsbbbfp=1cecffe8  bln=22  avl=00  flg=01
WAIT #3: nam='SQL*Net message to client' ela= 1 driver id=1111838976 #bytes=1 p3
=0 obj#=-1 tim=189900324244
EXEC #3:c=0,e=1079,p=0,cr=0,cu=0,mis=1,r=1,dep=0,og=1,plh=0,tim=189900324271
WAIT #3: nam='SQL*Net message from client' ela= 8281 driver id=1111838976
#bytes=1 p3=0 obj#=-1 tim=189900332625
```

# LUW bind into cached function

```
PARSING IN CURSOR #2 len=78 dep=0 uid=91 oct=47 lid=91 tim=189900332969 hv=
230108677 ad='7ffbf2b66d8' sqlid='782bwm86vfbh5'
BEGIN :status := sell_pkg.sell_and_ship2(:custid,:skucs,:purchase_qty); END;
END OF STMT
PARSE #2:c=0,e=143,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=0,tim=189900332968
BINDS #2:
 Bind#0
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=96 off=0
  kxsbbbfp=1cecffa0  bln=22  avl=00  flg=05
 Bind#1
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=24
  kxsbbbfp=1cecffb8  bln=22  avl=02  flg=01
  value=1
 Bind#2
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=48
  kxsbbbfp=1cecffd0  bln=22  avl=02  flg=01
  value=2
 Bind#3
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=72
  kxsbbbfp=1cecffe8  bln=22  avl=02  flg=01
  value=54
```

# LUW – fetch rowid (shortest for trans)

```
PARSING IN CURSOR #3 len=52 dep=1 uid=91 oct=3 lid=91 tim=189900333480 hv=
1354294563 ad='7ffbf5bc5c8' sqlid='dmf0kkj8bjt93'
SELECT I.ROWID FROM INVENTORY I WHERE I.SKUCS = :B1
END OF STMT
PARSE #3:c=0,e=34,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=1415135660,tim=
189900333479
BINDS #3:
 Bind#0
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1cecaa08  bln=22  avl=02  flg=05
  value=2
EXEC #3:c=0,e=93,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=1415135660,tim=
189900333644
FETCH #3:c=0,e=48,p=0,cr=1,cu=0,mis=0,r=1,dep=1,og=1,plh=1415135660,tim=
189900333733
STAT #3 id=1 cnt=1 pid=0 pos=1 obj=178147 op='INDEX UNIQUE SCAN INVENTORY_PK
(cr=1 pr=0 pw=0 time=0 us cost=0 size=8 card=1)'
CLOSE #3:c=0,e=3,dep=1,type=3,tim=189900333832
```

# LUW – check for enough via rowid

```
PARSING IN CURSOR #3 len=53 dep=1 uid=91 oct=3 lid=91 tim=189900333975 hv=
759153117 ad='7ffbf5bc1c8' sqlid='gywyzunqmzhfx'
SELECT I.ONHAND FROM INVENTORY I WHERE I.ROWID = :B1
END OF STMT
PARSE #3:c=0,e=31,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=4009304371,tim=
189900333975
BINDS #3:
 Bind#0
  oacdty=208 mxl=3950(3950) mxlc=00 mal=00 scl=00 pre=00
  oacflg=01 fl2=1206001 frm=00 csi=00 siz=3952 off=0
  kxsbbbfp=1cec7e50  bln=3950  avl=13  flg=05
  value=
Dump of memory from 0x000000001CEC7E50 to 0x000000001CEC7E5D
01CEC7E50 B7020001 000600E2 00010900 00000001  [................]
EXEC #3:c=0,e=103,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=4009304371,tim=
189900334141
FETCH #3:c=0,e=22,p=0,cr=1,cu=0,mis=0,r=1,dep=1,og=1,plh=4009304371,tim=
189900334194
STAT #3 id=1 cnt=1 pid=0 pos=1 obj=178146 op='TABLE ACCESS BY USER ROWID
INVENTORY (cr=1 pr=0 pw=0 time=0 us cost=1 size=8 card=1)'
CLOSE #3:c=0,e=1,dep=1,type=3,tim=189900334261
```

# LUW – gather customer facts

```
PARSING IN CURSOR #3 len=132 dep=1 uid=91 oct=3 lid=91 tim=189900334331 hv=
3497336310 ad='7ffbf5bbd68' sqlid='amv9xsb87a7gq'
SELECT C.NAME, C.SHIPTO_ADDR1, C.SHIPTO_ADDR2, C.SHIPTO_CITY, C.SHIPTO_STATE,
C.SHIPTO_POSTAL FROM CUSTOMER C WHERE C.CUST_ID = :B1
END OF STMT
PARSE #3:c=0,e=35,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=3389176309,tim=
189900334331
BINDS #3:
 Bind#0
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1cec8888  bln=22  avl=02  flg=05
  value=1
EXEC #3:c=0,e=79,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=3389176309,tim=
189900334475
FETCH #3:c=0,e=18,p=0,cr=2,cu=0,mis=0,r=1,dep=1,og=1,plh=3389176309,tim=
189900334523
STAT #3 id=1 cnt=1 pid=0 pos=1 obj=178148 op='TABLE ACCESS BY INDEX ROWID
CUSTOMER (cr=2 pr=0 pw=0 time=0 us cost=1 size=82 card=1)'
STAT #3 id=2 cnt=1 pid=1 pos=1 obj=178149 op='INDEX UNIQUE SCAN CUSTOMER_PK (cr=
1 pr=0 pw=0 time=0 us cost=0 size=0 card=1)'
CLOSE #3:c=0,e=1,dep=1,type=3,tim=189900334600
```

# LUW – grab transaction id we'll need

```
PARSING IN CURSOR #3 len=36 dep=1 uid=91 oct=3 lid=91 tim=189900334672 hv=
7117464 ad='7ffbf5bbb78' sqlid='8cums1w06t6ns'
Select SHIPPING_ID.NEXTVAL from dual
END OF STMT
PARSE #3:c=0,e=43,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=3236968432,tim=
189900334671
EXEC #3:c=0,e=30,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=3236968432,tim=
189900334761
FETCH #3:c=0,e=27,p=0,cr=0,cu=0,mis=0,r=1,dep=1,og=1,plh=3236968432,tim=
189900334820
STAT #3 id=1 cnt=1 pid=0 pos=1 obj=178154 op='SEQUENCE  SHIPPING_ID (cr=0 pr=0
pw=0 time=0 us)'
STAT #3 id=2 cnt=1 pid=1 pos=1 obj=0 op='FAST DUAL  (cr=0 pr=0 pw=0 time=0 us
cost=2 size=0 card=1)'
CLOSE #3:c=0,e=1,dep=1,type=3,tim=189900334897
```

- And re-check quantity available again using rowid
- Then *finally* do the transaction as quickly as we can!

# LUW – take our quantity via rowid

```
PARSING IN CURSOR #3 len=69 dep=1 uid=91 oct=6 lid=91 tim=189900335282 hv=
732238760 ad='7ffbf2b80b8' sqlid='fu3n8rspua4x8'
UPDATE INVENTORY I SET I.ONHAND = I.ONHAND - :B2 WHERE I.ROWID = :B1
END OF STMT
PARSE #3:c=0,e=30,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=1226990995,tim=
189900335281
BINDS #3:
 Bind#0
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1c703b38  bln=22  avl=02  flg=09
  value=54
 Bind#1
  oacdty=208 mxl=3950(3950) mxlc=00 mal=00 scl=00 pre=00
  oacflg=01 fl2=206001 frm=00 csi=00 siz=3952 off=0
  kxsbbbfp=1cec5658  bln=3950  avl=13  flg=05
  value=
Dump of memory from 0x000000001CEC5658 to 0x000000001CEC5665
01CEC5650                   B7020001 000600E2          [........]
01CEC5660 00010900 00000001                       [........]
EXEC #3:c=0,e=317,p=0,cr=1,cu=3,mis=0,r=1,dep=1,og=1,plh=1226990995,tim=
189900335662
STAT #3 id=1 cnt=0 pid=0 pos=1 obj=0 op='UPDATE  INVENTORY (cr=1 pr=0 pw=0 time=
0 us)'
STAT #3 id=2 cnt=1 pid=1 pos=1 obj=178146 op='TABLE ACCESS BY USER ROWID
INVENTORY (cr=1 pr=0 pw=0 time=0 us cost=1 size=8 card=1)'
CLOSE #3:c=0,e=0,dep=1,type=3,tim=189900335758
```

# LUW

```
PARSING IN CURSOR #3 len=205 dep=1 uid=91 oct=2 lid=91 tim=189900335822 hv=
4050659732 ad='7ffbf2da988' sqlid='dznwb0zsr0acn'
INSERT INTO SHIPPING S (S.SHIP_ID, S.SKUCS, S.QUANTITY, S.NAME, S.SHIPTO_ADDR1,
S.SHIPTO_ADDR2, S.SHIPTO_CITY, S.SHIPTO_STATE, S.SHIPTO_POSTAL) VALUES (:B9 ,
:B8 , :B7 , :B6 , :B5 , :B4 , :B3 , :B2 , :B1 )
END OF STMT
PARSE #3:c=0,e=30,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=0,tim=189900335822
BINDS #3:
 Bind#0
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1c703cc0  bln=22  avl=02  flg=09
  value=3
```

- And a bunch more binds we have already have values for

```
Bind#8
  oacdty=01 mxl=128(100) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=01 csi=873 siz=128 off=0
  kxsbbbfp=1c704000  bln=128  avl=10  flg=09
  value="03766-1239"
EXEC #3:c=0,e=559,p=0,cr=1,cu=8,mis=0,r=1,dep=1,og=1,plh=0,tim=189900336451
STAT #3 id=1 cnt=0 pid=0 pos=1 obj=0 op='LOAD TABLE CONVENTIONAL  (cr=1 pr=0 pw=
0 time=0 us)'
CLOSE #3:c=0,e=1,dep=1,type=3,tim=189900336524
=====================
PARSING IN CURSOR #3 len=6 dep=1 uid=91 oct=44 lid=91 tim=189900336561 hv=
255718823 ad='0' sqlid='8ggw94h7mvxd7'
COMMIT
```

# Reviewing LUW

- Used a stored procedure
  - ship minimum code(could encode names, too)
  - fetched clean blocks, values
  - rowids for re-fetch and update single rows
  - bail without starting transaction if need be
  - once transaction starts there is no fat
- Package was warmed up in shared pool
  - can build a relatively static pool of “always there”
  - trade-off between size and possible transaction time load

# Reviewing LUW

- Showed a few “extra work” for smaller footprint bits
  - Initial package would have been simpler
  - fetched clean blocks, values
  - rowids for re-fetch and update single rows
  - bail without starting transaction if need be
  - would be reactions to needing the smaller footprint
  - are extra total work, but outside “transaction time”
- So this was partially beyond lazy rule for demo

# On beyond zebra

- Beneath package you might manipulate storage
  - Hermitize extremely popular inventory items
  - Artificial “virtual” warehouse
    - partition
    - might require inventory re-balance
  - Surround in block with dummy items

- These are examples for the inventory/sales metaphor.

OakTable World

DOAG

# Minimizing the Concurrency Footprint of Transactions

Mark W. Farnham

Rightsizing, Inc.

RIGHTSIZING INC

Oracle Database Technology and Business Consulting

OakTable®