Americas PUG Challenge 2014

Westford, MA June 8 - 11, 2014

2393: What's that user doing ?

Gus

Westford, MA June 8 - 11, 2014

"Lock table overflow, increase –L on server"

User sessions freeze and promon shows something like this:

```
12/07/10          Status: Lock Table

 Usr  Name  Trans ID  Type  Rec-id  Table  Flags  Tran  State
   6  user1   121759   REC    8199      2      X        Begin
   7  user2   121761   REC    8199      2    S Q        Begin
   7  user2   121761   REC    9000      5      X        Begin
   6  user1   121759   REC    9000      5  S QH         Begin
```

Fear not !
There are ways.

Generating 4GL call stack files:

Manually Generate Stack Trace

Enhanced protrace format

- Stack also available via ABL dump request
  - Useful for “non-responsive” connections
  - Available with 10.1c
  - Must have access to client’s machine
- kill –SIGUSR1 <pid> (Don’t forget the dash)
  - protrace.<pid>
    - Startup parameters
    - Execution stack (Statement cache)
    - ** ABL Stack Trace **
    - ** Persistent procedures/Classes **
    - ** PROPATH **
    - ** Databases (logical/type/physical) **

NOT for SQL clients !

Generate call stack on Windows

```
proGetStack <pid>
```

4GL call stack added in 10.1C

The client database-request statement cache:

Client Database-Request Statement Cache

- What ?
  - 4GL procedure call monitor for db requests
  - Line #, procedure name, file name
  - Not new: introduced in OpenEdge release 10.1C
- Why ?
  - Diagnosing application problems
- How?
  - Documentation: *OpenEdge Data Management: Database Administration*
  - We will tell you how today

The 4GL Procedure Call Stack

A LIFO list …

- Top: most recent procedure/function/method call
- Bottom: oldest

```
  #    Procedure Name                                File Name

 19  : reallyLongNamedInternalProcedure3   proctestb.r
 12  : reallyLongNamedInternalProcedure2   proctestb.r
  5  : reallyLongNamedInternalProcedure1   proctesta.r
445  : reallyLongNamedInternalProcedure0   proctesta.r
  1  : /usr1/stmtest/p72340_Untitled1.ped
```

- More on data format later…

How you turn it on then ?

In promon R&D,
1. Status Displays,
18. Client Database-Request Statement Cache

```
12/10/07 OpenEdge Release 10 Monitor (R&D)
09:06:10 Client Database-Request Statement Caching Menu

    1. Activate For Selected Users
    2. Activate For All Users
    3. Activate For All Future Users
    4. Deactivate For Selected Users
    5. Deactivate For All Users
    6. Deactivate For All Future Users
    7. View Database-Request Statement Cache
    8. Specify Directory for Statement Cache Files
```

Activation Types – 1 of 3

- 1. Single (procedure)
  - Limited information, least overhead
  - Last procedure call only
  - Updated when new db operations occur

Activation Types – 2 of 3

- 2. Stack
  - Reports entire stack (31 deep, sometimes more)
  - 32,000 byte maximum stack size
  - Most information
    - How did I get there?
  - Updated when new db operations occur

Activation Types – 3 of 3

- 3. One time stack
  - Full stack snapshot
  - Reports stack one time only (on next DB operation)
  - Not updated
    - Remembered until deactivated or reactivated
    - Re-activate for update
  - Useful when things change too fast

Turning it on for a server

Multiple clients on same server.

- Server level enablement
  - Activates all currently served clients
  - Remote logout/login disables it
- Application server agent
  - Connection based, not session based
  - Setting does not follow user through to AppServer Agent
- OpenEdge SQL Server
  - Same activation rules
  - Statement level report
  - Stack level does not apply
  - Updating applies

How you turn it off ?

In promon R&D,
1. Status Displays,
18. Client Database-Request Statement Cache

```
12/10/07 OpenEdge Release 10 Monitor (R&D)
09:06:10 Client Database-Request Statement Caching Menu

      1. Activate For Selected Users
      2. Activate For All Users
      3. Activate For All Future Users
      4. Deactivate For Selected Users
      5. Deactivate For All Users
      6. Deactivate For All Future Users
      7. View Database-Request Statement Cache
      8. Specify Directory for Statement Cache Files
```

How you turn it off ?

- Deactivate server deactivates all users of that server.

12/07/10 OpenEdge Release 11 Monitor (R&D)
11:15:03 Deactivate For Selected Users

Usr	Name	Type	Login time	Serv	IPV#	Remote Address
1	pug11	SERV	12/06/10 16:19	0		
23	usera	REMC/ABL	12/06/10 16:23	1	IPV4	172.12.3.456
24	userb	REMC/ABL	12/07/10 08:34	1	IPV4	172.12.3.456

4. Deactivate For Selected Users
5. Deactivate For All Users
6. Deactivate For All Future Users

How you look at the data ?

In promon R&D,
1. Status Displays,
18. Client Database-Request Statement Cache

```
12/10/07 OpenEdge Release 10 Monitor (R&D)
09:06:10 Client Database-Request Statement Caching Menu

      1. Activate For Selected Users
      2. Activate For All Users
      3. Activate For All Future Users
      4. Deactivate For Selected Users
      5. Deactivate For All Users
      6. Deactivate For All Future Users
      7. View Database-Request Statement Cache
      8. Specify Directory for Statement Cache Files
```

3 Caching Types

12/07/10 OpenEdge Release 11 Monitor (R&D)
11:20:50 View Database-Request Statement Cache

Usr	Name	Type	Login time	Serv	Type	Cache Update
5	userb	SELF/ABL	12/07/10 09:38	0	L1	12/07/10 09:50
23	userb	REMC/ABL	12/07/10 09:46	2		
24	userb	REMC/ABL	12/07/10 09:40	1	L2	12/07/10 09:50
25	userb	REMC/ABL	12/07/10 09:40	1	RQ	12/07/10 09:50
26	userb	REMC/SQL	12/07/10 09:40	3	L2	12/07/10 09:50

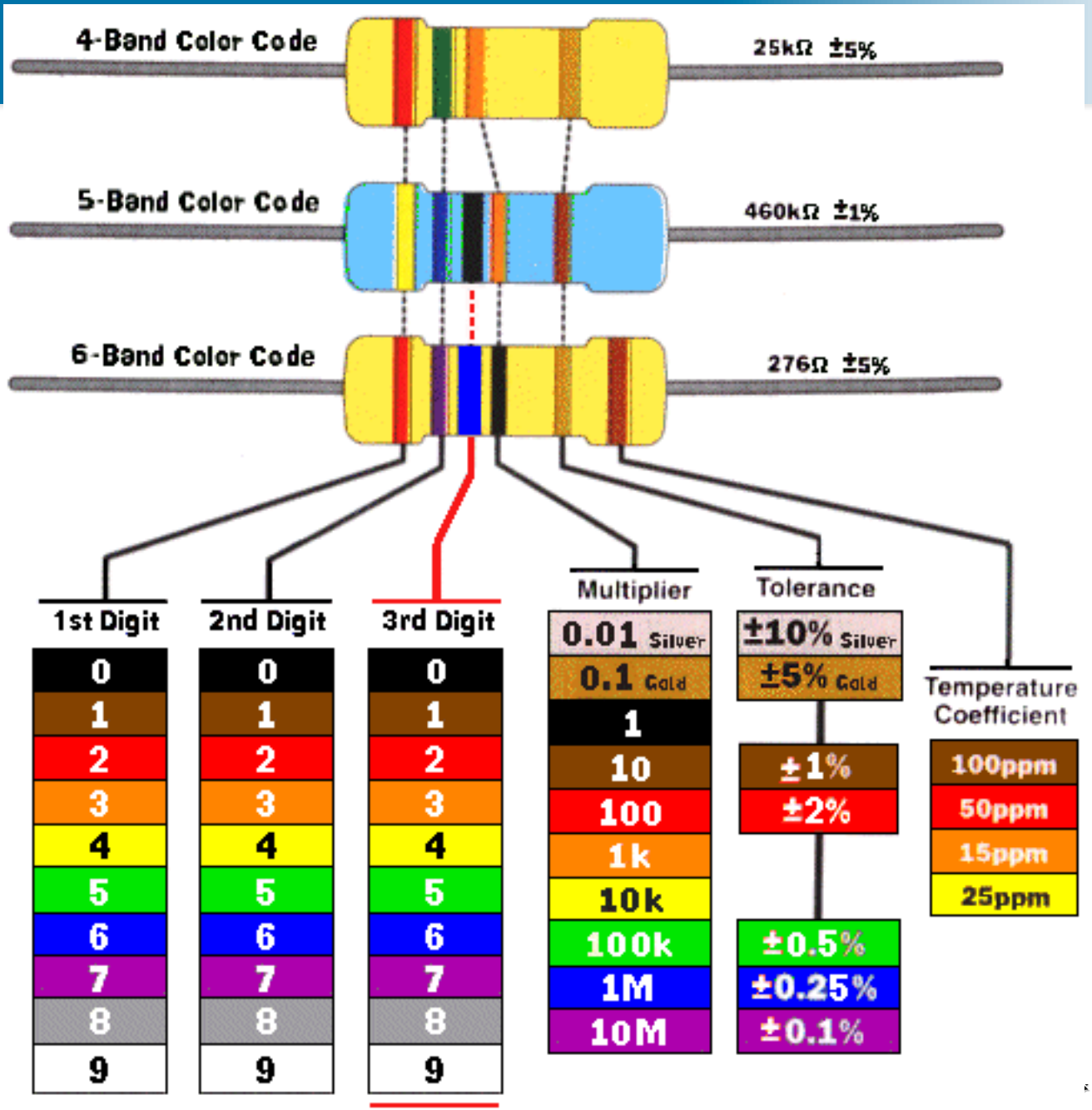
Type

- Describes value in the cache
  - L1, L2, RQ
- Remains blank if cache not populated yet

3 Caching Types

- "L1": Level 1
  - Updated procedure at top of stack only
  - *"SQL Statement or Single ABL Program Name"*
- "L2": Level 2
  - Updated full stack
  - *"SQL Statement or ABL Program Stack"*
- "RQ": Client Trace Request (level 3)
  - One time stack
  - *"SQL Statement or ABL Program Stack"*

32,000 BYTES DATA LIMIT



SQL Statements

```
12/07/10        OpenEdge Release 11 Monitor (R&D)
11:25:34        View Database-Request Statement Cache

User number                    : 22
User name                      : userb
User type                      : REMC/SQLC
Login date/time                : 12/07/10 09:02
Statement caching type         : SQL Statement or Partial ABL Program Stack
Statement caching last updated : 12/07/10  11:23

Statement cache information    : select count(*) from pub.customer
```

- Operates on user or server level - same rules apply
  - "All users" option does not indicate server specifically
- Lower overhead
  - Statement level entry
  - No additional network traffic (all server side)
- Line number not used (always 0 in VST)

ABL Statements

```
12/07/10        OpenEdge Release 11 Monitor (R&D)
11:25:34        View Database-Request Statement Cache

...
Statement cache information     :
                  2 : finalProcedure   userb/my_dot_rs/longNamedDotr4.r
                  4 : userb/my_dot_rs/longNamedDotr3.r
                  4 : userb/my_dot_rs/longNamedDotr2.r
                  4 : userb/my_dot_rs/longNamedDotr1.r
                  1 : r2.r
```

- Line #: procedure/function name, file name
  - Maps to COMPILE w/DEBUG-LIST option
  - Pathname same as in run statement

ABL Statements

```
12/07/10        OpenEdge Release 11 Monitor (R&D)
11:25:34        View Database-Request Statement Cache

...
Statement cache information     :
                    2 : finalProcedure   userb/my_dot_rs/longNamedDotr4.r
                    4 : userb/my_dot_rs/longNamedDotr3.r
                    4 : userb/my_dot_rs/longNamedDotr2.r
                    4 : userb/my_dot_rs/longNamedDotr1.r
                    1 : r2.r
```

- When no internal procedure or function, just line # and file name

Statement Caching for OO ABL

```
12/07/10        OpenEdge Release 11 Monitor (R&D)
11:25:34        View Database-Request Statement Cache

...
Statement cache information     :
                  17 : reallyLongNamedProcedure3   proctest2.p
                  12 : reallyLongNamedProcedure2   proctest2.p
                   5 : reallyLongNamedProcedure1   proctest2.p
                 445 : proctest2.p
                  16 : methodB     test13d
                   3 : runner.p
                   1 : /usr1/userb/11/stmtest/p49070_Untitled1.ped
```

- Line #: Method name
- Class file name without .cls extension

Statement Caching for OO ABL

```
12/07/10       OpenEdge Release 11 Monitor (R&D)
11:25:34       View Database-Request Statement Cache

...
Statement cache information     :
                  17 : reallyLongNamedProcedure3   proctest2.p
                  12 : reallyLongNamedProcedure2   proctest2.p
                   5 : reallyLongNamedProcedure1   proctest2.p
                 445 : proctest2.p
                  16 : methodB     test13d
                   3 : runner.p
                   1 : /usr1/userb/11/stmtest/p49070_Untitled1.ped
```

- Code that was run from proc editor (note .ped)

About line number -1 ...

```
12/07/10        OpenEdge Release 11 Monitor (R&D)
11:25:34        View Database-Request Statement Cache

...
Statement cache information     :
                  -1 : reallyLongNamedProcedure1   proctest2.p
                 445 : proctest2.p
                  16 : methodB     test13d
                   3 : runner.p
                   1 : /usr1/userb/11/stmtest/p49070_Untitled1.ped
```

- Line number "-1" if
  - Database action at end of procedure
  - Not a specific line # in a .p
  - Often the result of buffer flushing

Specify Scratch File Directory

In promon R&D,
1. Status Displays,
18. Client Database-Request Statement Cache

```
12/10/07 OpenEdge Release 10 Monitor (R&D)
09:06:10 Client Database-Request Statement Caching Menu

      1. Activate For Selected Users
      2. Activate For All Users
      3. Activate For All Future Users
      4. Deactivate For Selected Users
      5. Deactivate For All Users
      6. Deactivate For All Future Users
      7. View Database-Request Statement Cache
      8. Specify Directory for Statement Cache Files
```

About Scratch Files

- One file per db connection
- Created when call stack data > 256 bytes
- Removed at disconnect
- Where ?
  - Default is .db directory
  - Sorry, not startup parameter
  - Change location via promon
  - New scratch files go to new location
- Generated file names
  - <fulldatabasename>.<pid>.<usrnum>.cst
  - “/” replaced by “~” in generated names
    usr1~foo~x.26106.5.cst OR C!~users~foo~x.40744.6.cst

Application Server

- Establish logging by user (AppServer Agent)
  - Same as any other user
- Tracing back
  - Actions based on AppServer agent connection
  - Connection/tracing maintained across session disconnect
  - Difficult to identify originating user request
- asbroker1.server.log
  - Contains procedure call info too!
    - Even at loggingLevel=1 (error only)

Examine call stacks using virtual system tables

_Connect VST 1

- _Connect-CachingType
  - Caching level: 01, 02, 03
  - Value “requested” in promon (top, stack, one-time)
- _Connect-CacheInfoType
  - "ABL Program", "SQL Statement", "ABL Stack“
  - Value of current stack type displayed
  - “?”: stack requested and no stack yet
- "ABL Program“ (01)
  - Procedure and .p name displayed
- "ABL Stack“ (02 & 03)
- “SQL Statement” (01, 02 & 03)

_Connect VST 2

- _Connect-CacheLastUpdate
  - Date/time of cache update
  - One time stack - indicates age of information
  - Updating stack – time of last database request
- _Connect-CacheInfo[32]
  - Up to “last“ 31 stack entries
  - Procedure & executing image name
    - .p or .r executed (run)
    - Pathname specified (not fully qualified)
- _Connect-CacheLineNumber[32]
  - Up to “last” 31 stack entries
  - Line number of code

Example _Connect VST Query

FAIL !

```
FOR EACH _Connect WHERE
        _Connect-CachingType <> ?:
```

“one-time” sets CachingType to “?” after processed.

Pass !

```
FOR EACH _Connect WHERE
        _Connect-CacheInfoType <> ?:
```

CacheInfoType set when data exists, not pending.

```
DISPLAY _Connect-id _Connect-Usr
        _Connect-CachingType
        _Connect-CacheInfoType format "x(12)“
        _Connect-CacheLineNumber[1]  label "Line“
        _Connect-CacheInfo[1]            label "Entry“
        _Connect-CacheLineNumber[2] no-label
        _Connect-CacheInfo[2]            no-label
END.
```

About VST row numbering ...

- VSTs have an "index" or VSI
  - Find by VSI field is usually quicker (for queries with where clause)
  - Index find vs table scan
  - Depends if table based or not
    - Lock table is chain based
  - Value always known
- VSI Counts from 1
- Data may count from 1 or 0 _Connect-user counts from 0

```
FIND _Connect WHERE
     _Connect-Id = aUserId + 1.
```

```
FOR EACH _Connection:
   DISPLAY _Connect-id
           _Connect-Usr
END.
```

Connect-Id	Connect-usr
1	0
2	1
3	?
4	?
5	?
6	5
7	?
8	?
9	?
10	9
11	10

statement cache memory requirements

Memory Consumption

- Statement cache is always “server side”
- Static allocation
  - 28 bytes * (–n + -Mn + 2), regardless of cache state
- Dynamic allocation
  - 288 bytes per enabled user
  - Allocated upon stack population
  - 32 bytes control info, 256 bytes actual stack
  - SHM reserved at DB startup
    - 80 bytes * (-n + -Mn + 2)
    - -Mxs is used for overflow
- Consumption subject to change without notice (internal data structures can change size in next release)

Effects on performance

Performance: Communication

- Client message coupled w/statement cache message
  - Each database “update” and find request
  - Each database “lock” request
    - (SHARE/EXCL)
    - Record/Schema
  - Same stack, no new message
  - Can more than double the network traffic
- Server side queries
  - Do not require additional messages
  - Client stack is not changing anyway!
- Full stack displayed if enabled while in use

Performance: Communication

- Client message coupled w/statement cache message
  - Each database “update” and find request
  - Each database “lock” request
    - (SHARE/EXCL)
    - Record/Schema
  - Same stack, no new message
  - Can more than double the network traffic
- Server side queries
  - Do not required additional messages
  - Client stack is not changing anyway!
- Full stack displayed if enabled while in use

Performance: Communication

- Client message coupled w/statement cache message
  - Each database “update” and find request
  - Each database “lock” request
    - (SHARE/EXCL)
    - Record/Schema
  - Same stack, no new message
  - Can more than double the network traffic

```
1:  for each customer no-lock:
2:      the_addr = address.
3:  end.
```

```
1:  for each customer share-lock:
2:      the_addr = address.
3:  end.
```

- Full stack displayed if enabled while in use

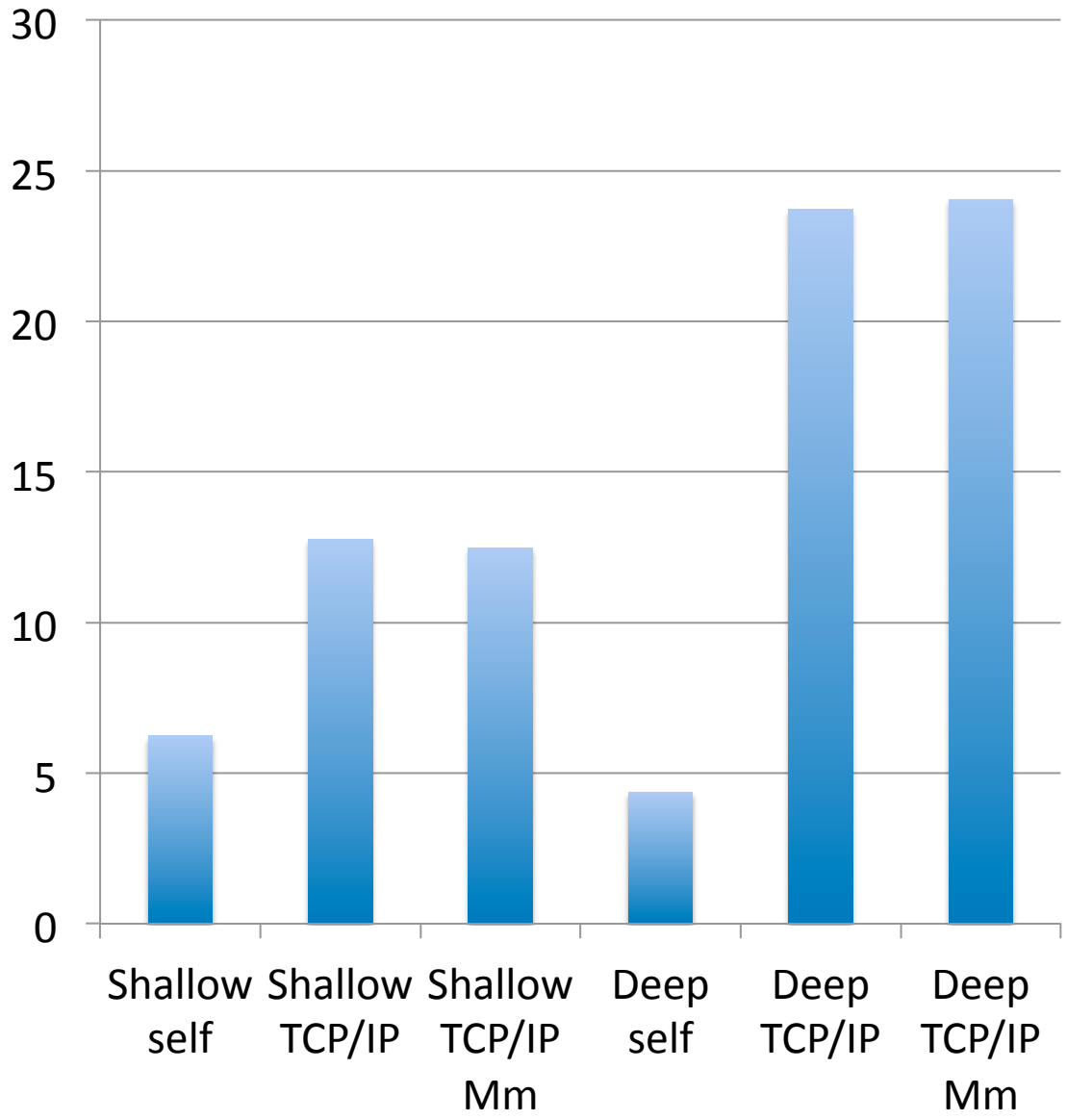
Performance: Communication

- Network message size limited by –Mm
  - If exceeded, multiple messages required
- How to get really large messages
  - Long .p names (run path) for one!
  - Deep stack for another.
- Help!
  - Increase –Mm (client AND server)
  - PROPATH to decrease “run” path length
  - Shorter function/procedure/file names
  - One-time request
  - Request stack top entry only
  - Be frugal with activation

Performance Impact

- Performance vs information trade off
  - Continuous full stack reporting
  - Modular programming
    - Improves information
    - Increases stack size
- Let's look at some data

DB-request logging performance



deep: about 3k of stack data

Mm: set to 8k has little effect

Self: stack depth has little effect

Scratch file I/O not a problem

YMMV

Back to the beginning of our talk:

Has this ever happened to you?

User sessions freeze and promon shows this:

```
12/07/10          Status: Lock Table

 Usr  Name Trans ID Type Rec-id Table Flags Tran State
   6  user1  121759  REC   8199     2     X       Begin
   7  user2  121761  REC   8199     2   S Q       Begin
   7  user2  121761  REC   9000     5     X       Begin
   6  user1  121759  REC   9000     5  S QH       Begin
```

Deadlock

User 1 waiting for user 2 to release X lock.

User 2 waiting for user 1 to release X lock.

```
12/07/10          Status: Lock Table

 Usr  Name Trans ID Type Rec-id Table Flags Tran State
   6  user1  121759  REC   8199     2     X       Begin
   7  user2  121761  REC   8199     2   S Q       Begin
   7  user2  121761  REC   9000     5     X       Begin
   6  user1  121759  REC   9000     5  S QH       Begin
```

Deadlock case 1

- Statement cache already enabled
  - View users in deadlock

```
12/07/10        OpenEdge Release 11 Monitor (R&D)
11:45:34        View Database-Request Statement Cache

Statement cache information     :
                22 : find-orders            x2.p
                20 : process-customers      x1.p
                25 : x.p                    x.p
```

```
12/07/10        OpenEdge Release 11 Monitor (R&D)
11:45:34        View Database-Request Statement Cache

Statement cache information     :
                32 : find-customers         c2.p
                30 : process-orders         c1.p
                35 : c.p                    c.p
```

Deadlock case 2

- Statement cache NOT already enabled
  - Enable and wait for next occurrence
- Too late to enable statement caching
  - Use *kill –SIGUSR1 <pid>*
  - *cat protrace.<pid>*

```
** ABL Stack Trace **

--> find-orders x2.p (x2.p) at line 22
    process-customers x1.p (x1.p) at line 20
    x.p (x.p) at line 25
```

```
** ABL Stack Trace **

--> find-customer c2.p (c2.p) at line 32
    process-orders c1.p (c1.p) at line 30
    c.p (c.p) at line 35
```

"Lock table overflow, increase –L on server"

Lock Table Overflow

- Find user taking all locks
  - could be doing a whole-index table scan
- Find bad code
- First, look at promon lock activity

```
12/07/10          Other: Lock Requests By User

Usr  User   --- Record ---  ---- Trans ---  --- Schema ---
     Name   Locks     Waits Locks     Waits Locks     Waits
  0  pug1       0         0     0         0     0         0
  5 userb      10         1     0         0     0         0
  6 userb    3213         1     0         0     0         0
```

Lock Table Overflow

- Enable statement cache for user

```
12/07/10          OpenEdge Release 11 Monitor (R&D)
11:45:34          View Database-Request Statement Cache

Statement cache information:
           445: report-customers       all_customers.p
            16: change-cust-address    update_customer.p
             3: manage-cust-acct       runner.p
```

- Track data back to user/code
- Get programmer to fix

Easy. Fast. Cheap.

Summary

- Database request cache is a "Cool Tool"
- Flexible
- Very useful, but has some performance impact
- Can be a razor blade

Also have a look at the manual:

OpenEdge Development: Debugging and Troubleshooting

any questions