Top Tips learned from Performance PMRs

Dr. Elisabeth Bach

IBM

D01

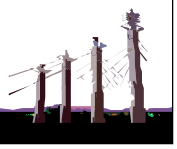
Monday, April 28, 2008 • 09:30 a.m. – 10:30 a.m.

2008 IIUG Informix Conference

www.iiug.org

Contents

- Performance PMRs -- a real world example
- How slow is too slow? Get a baseline
- What is your application doing anyway? SQL and other tracing
- A few new onstats
- Build my index faster!
- Update statistics and PDQ
- Top Tips Summary



Stories from support

- It was on one Monday morning…..
- Everything was slow
- We needed to fix it IMMEDIATELY

What has happened?

What was changed?

What is bad now?

Rules of thumb

You cannot extrapolate from one measurement.
Always repeat a few times.

Where there is a persistent queue, there is likely a bottleneck.

Look at the online.log

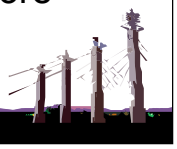
Stick to your processes, especially during times of stress.

DONTDRAINPOOLS

- Environment variable
  Set DONTDRAINPOOL=1
  Read during initialization
- Writes a message to the online.log

```
Server is disabling pools draining
```

- Sessions might use more memory as before



DONTDRAINPOOLS is an environment variable that influences the way memory allocated to IDS user sessions is de-allocated. The default behavior of the database server is to aggressively drain unused memory from a session's pool .This default setting make sense, if sessions are around for some time, if they request and free memory often and if the amount of memory is limited.

In an environment in which we have many sessions, and performance is more likely to be cpu bound than memory bound, it can be useful to set DONTDRAINPOOLS. It is also useful in environments like that of our customer, which is using many cpu-vps and a greater number of sessions.

Setting DONTDRAINPOOLS mean that the freeing of unused memory of user sessions is done only at closure time.

This can result in the server utilizing more memory overall, however this might be more desirable than the high CPU usage.

This means that the cpu will have less work to do, however every single session can now use more memory.

How much more memory is used is dependent on the individual system and application and has to be tested.

How slow is too slow?

- What is your system doing anyway?
- How to monitor a perfectly well-behaved system
  - Shell script for UNIX
  - Scripts for Windows
  - OAT

BASELINE INFO IS ESSENTIAL FOR PERFORMANCE TUNING

Snooping at your system

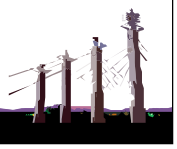
System information

- cpu usage
- memory usage
- disk usage

Onstats

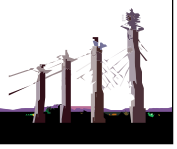
- onstat –g glo
- onstat –g rea
- onstat –g act

what else is dependent on your system



Snooping at your system: UNIX

- Use system tools for system info
  - sar, top, vmstat
  - What is available
- You can use a never ending script or schedule it
- Find an example in the notes

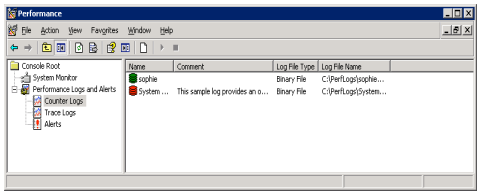


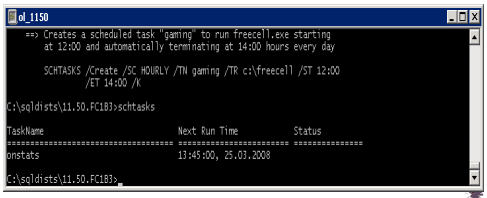

```
#!/usr/bin/ksh
#
#               Seconds to sleep between next monitoring loop
#               e.g. GENERAL_SLEEP=3600
#               sleep 3600 seconds between measurements
#
GENERAL_SLEEP=3600
REA_SLEEP=2
SPI_SLEEP=2
#               Endless:
while [ 1 -ge 0 ]
do
                UHRZEIT=$(date +%Y-%m-%d-%H:%M:%S)
                for g in d D p
                do
                                onstat -$g > onstat-$g-----$UHRZEIT.out
                done

                UHRZEIT=$(date +%Y-%m-%d-%H:%M:%S)
                for g in glo seg iof mem dic
```

Snooping at your system: Windows

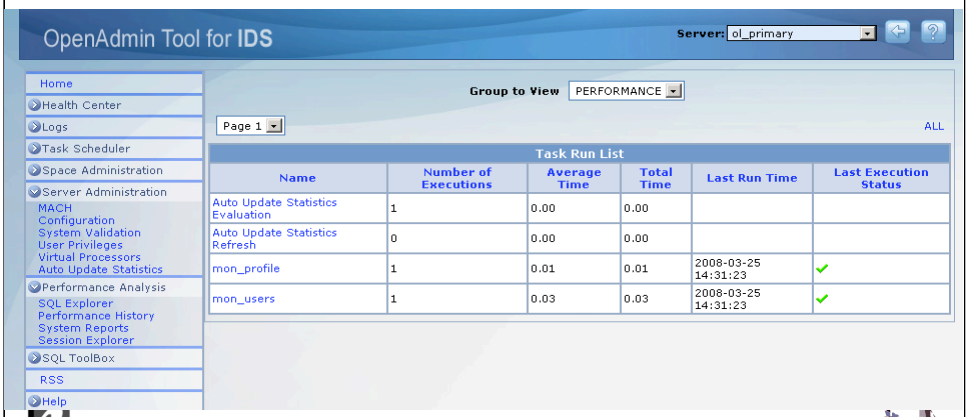
- Use Performance Monitor's report functionality for system information
- Consider using WMI
- Use schtasks for scheduling
- Find a script in the notes






```
@echo off
rem
rem these variable are set so that we can append the date and time
rem if you have an other date delimiter than “.” you need to do the same thing
rem for date as for time
Rem
set startDate=%date%
set startTime=%time%
set /a sth=%startTime:~0,2%
set /a stm=1%startTime:~3,2% - 100
set /a sts=1%startTime:~6,2% - 100
set onsttime=%startDate%.%sth%.%stm%.%sts%
rem set informix env here
rem
set INFORMIXDIR=C:\sqldists\1150~1.FC1
set INFORMIXSERVER=ol_1150
set ONCONFIG=ONCONFIG.ol_1150
set PATH=C:\sqldists\1150~1.FC1\bin;%PATH%
set
CLASSPATH=%INFORMIXDIR%\extend\krakatoa\krakatoa.jar;%INFORMI
XDIR%\extend\krakatoa\idbc.jar;%CLASSPATH%
```

Snooping at you system with OAT

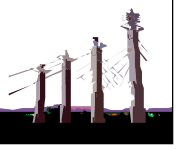


The tasks introduced in version 11.10 add the ability to continually monitor the system. You can define jobs to gather monitoring information, some jobs are automatically defined.

OAT gives you the ability to see those tasks, to change their execution frequency and with OAT 2.20 new graphics to display this information have been added.

What is my application doing?

- Application developer and DBA are different jobs
- Applications might be developed using tools that don't tell you the queries they are using
- What to do when the users are complaining?

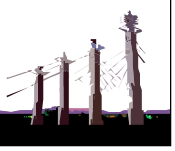


SQL tracing before 11.10

- Before version 11.10 SQL tracing was not easy
- So how to find out which statement is used often?

- Onstat –g sql shows session info
- Only a snapshot

- Gather data often and compare statements
- Find a script in notes (run as informix!)

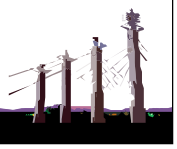



```
#!/usr/bin/ksh
if [ ! -x /usr/bin/ksh ]
        then
        echo "please change '#!/usr/bin/ksh' in line 1 to the path of a ksh"
        echo "compatible shell and delete this if ..."
        exit 1
fi
# Multiuser OnLine sql statistics without sysmaster.
#
# Usage: SqlStat nsamples sleep-in-secs
#
# Results are in directory "erg.<hh:mm:ss>"
#
# Filename: erg.<hh:mm:ss>/0...0<num>.*_*
# <num> is the number of samples for the sql statement in this file.
# Just use "ls -l" to show files ordered by fequency of sql statements.
#
nsamples=120
sleep=1
[ $# -ge 1 ] && nsamples="$1"
```

What you get with the script

- 00005.2935_1 statement 2935 was found 5 times
- 00008.2859_1 statement 2859 was found 8 times
- 00007.8876_1 statement 8876 was found 7 times
- INFO shows how many statements were found
- Each files shows
  - Statement: create table t1 (c1 int, c2 char(20))
  - What statement was doing

    @ 31.03.15:02, ses 2257, user ebach, pid 22, host 10904, fe cond wait cp 1cpu sqlexec

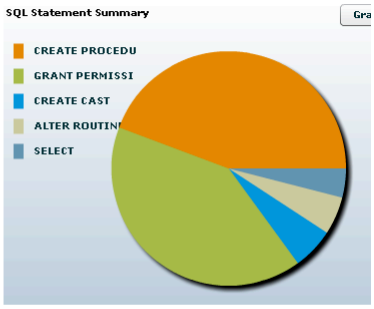


SQLtrace with 11.10

Statement Type | Transaction Time | Frequency | SQL Tracing Admin

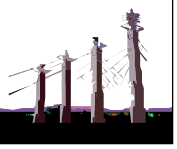
SQL Statement Summary

Statement Type	Count	Avg Response Time	Max Response Time	Avg Memory	Rows Processed
CREATE PROCEDU	799	0.0019	0.0832	11.1 KB	0
GRANT PERMISSI	735	0.0001	0.0310	5.04 KB	0
CREATE CAST	104	0.0000	0.0000	4.19 KB	0
ALTER ROUTINE	94	0.0005	0.0212	3.94 KB	0
SELECT	72	0.0302	0.4839	37.1 KB	146
CREATE OPAQUE TYPE	34	0.0000	0.0000	3.00 KB	0
INSERT	29	0.0026	0.0244	8.35 KB	2
CREATE TABLE	27	0.0873	0.7128	7.06 KB	0
DROP CAST	24	0.0000	0.0001	3.76 KB	0
CREATE AGGREGATE	18	0.0000	0.0001	4.01 KB	0
CLOSE DATABASE	9	0.0000	0.0000	2.00 KB	0
CREATE DISTINCT TYPE	9	0.0001	0.0002	5.01 KB	0
DATABASE	6	0.0003	0.0003	3.00 KB	0
SET LOCK MODE	6	0.0000	0.0000	3.00 KB	0
CREATE ACCESS_METHOD	6	0.0000	0.0000	11.7 KB	0
ALTER ACCESS_METHOD	4	0.0000	0.0001	4.00 KB	0



What can you do?

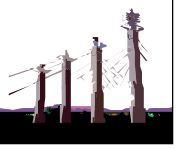
- Change the application
- If you can‘t, check the query plan
  - Check update statistics currency
  - Add additional indexes
  - Use optimizer hints
  - If you can‘t
    - Consider using external directives



onstats

You cannot extrapolate from one measurement.
Always repeat a few times.

- New or improved in 11.10
  - onstat –g cpu
  - onstat –g ath
  - onstat –g ckp



onstat –g cpu

- New with version 11.10
- Tells you when each thread was scheduled
- Identifies what is currently hanging
- Use along with onstat –g stk to see what threads are doing/waiting for

```
IBM Informix Dynamic Server Version 11.10.FC2W1   -- On-Line -- Up 3 days 05:51:27 -- 8649728 Kbytes

Thread CPU Info:
 tid     name              vp        Last Run          CPU Time     #scheds   status
  4   aio vp 0          36aio*     03/26 17:41:55      491.8493      74091   IO Idle
  5   msc vp 0          37msc*     03/26 17:41:59     8544.2635   10100529   IO Idle
  6   aio vp 1          38aio*     03/26 17:12:08        5.8684        426   IO Idle
  7   main_loop()       13cpu      03/26 17:41:59      244.5516     576872   sleeping secs: 1
  8   tlitcppoll        39tli*     03/26 17:41:59   280247.3047  138388591   running
 15   flush_sub(0)      13cpu      03/26 17:41:59      121.2926    1401344   sleeping secs:
 16   flush_sub(11)     7cpu       03/26 17:41:58       66.5788    1044132   IO Wai
 26   kaio              1cpu*      03/26 17:41:59   141010.0865   44337017   running
 52   kaio              3cpu*      03/26 17:41:59   135401.9964   40585737   IO Idle
 80   kaio              7cpu*      03/26 17:41:59   115871.8548   28019443   running
138   aio vp 4          47aio*     03/23 11:51:36        0.3423         44   IO Idle
139   aio vp 5          48aio*     03/23 11:51:48        0.4359         74   IO Idle
140   aio vp 6          49aio*     03/23 11:52:11        0.7263         80   IO Idle
```

onstat –g ath

- Now comes with more detailed information
- No longer simply „sleeping“

```
IBM Informix Dynamic Server Version 11.10.FC2W1   -- On-Line (CKPT REQ) -- Up 15 days 00:36:28 --
     197632 Kbytes
Blocked:CKPT

Threads:
 tid     tcb              rstcb            prty status                vp-class       name
 2       1113a1ad8        0                1    running               3lio*          lio vp 0
 3       1113c1c80        0                1    IO Idle               4pio*          pio vp 0
 4       1113e0c80        0                1    running               5aio*          aio vp 0
 5       1113ffc80        0                1    IO Idle               6msc*          msc vp 0
 6       11142ec80        0                1    IO Idle               7aio*          aio vp 1
 7       1113a1d28        1111c9028        1    sleeping secs: 1      11cpu          main_loop()
 40      1116ca348        1111ce930        1    sleeping forever      1cpu*          dbWorker2
 48      111f83b58        1111cc8d0        1    cond wait  bp_cond    9cpu           bf_priosweep()
 242     11224c028        0                1    IO Idle               18aio*         aio vp 2
 336     1126e18e8        1111d19c0        1    cond wait  cp         1cpu           sqlexec
 338     11505fbf8        1111e6db0        1    IO Wait               9cpu           sqlexec
 339     1126c8028        1111e4d50        1    IO Wait               1cpu           sqlexec
```

onstat –g ckp

- Information about the last 15 checkpoints
- Check for what is triggering the checkpoint, duration, blocking time and flush rates to get an idea of what the system is doing

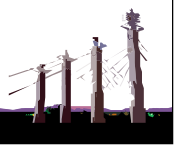
	Clock			Critical Sections						Physical Log		Logical Log	
Interval	Time	Trigger	LSN	Total Time	Flush Time	Block # Time	Waits	Ckpt Time	Wait Time	Long Time	# Dirty Buffers	Dskflu /Sec	Total Pages	Avg /Sec	Total Pages	Avg /Sec
972	15:47:45	CKPTINTVL	1202645:0x1a2c8018	80.9	54.0	0.0	106	0.9	19.4	26.0	88184	1632	11968	35	17988	53
973	15:47:46	*Backup	1202645:0x1b3fb3a4	0.5	0.3	0.0	12	0.0	0.5	0.5	4867	4867	4396	79	4403	80
974	15:54:47	CKPTINTVL	1202646:0x2516064	92.9	87.7	0.0	70	0.6	3.3	4.4	78129	890	28595	85	29965	89
975	15:58:40	CKPTINTVL	1202646:0xc209018	21.6	20.6	0.0	25	0.0	0.8	0.9	36483	1768	37922	126	41051	136
976	16:03:55	CKPTINTVL	1202646:0x14792018	12.3	11.4	0.0	34	0.0	0.5	0.8	39396	3448	33283	102	34913	107
977	16:09:16	CKPTINTVL	1202647:0x2098018	22.0	21.8	0.0	11	0.0	0.1	0.2	43636	2004	56345	180	53651	171

Max Plog pages/sec	Max Llog pages/sec	Max Dskflush Time	Avg Dskflush pages/sec	Avg Dirty pages/sec	Blocked Time
3207	2222	1741	2611	14	0

Based on the current workload, the physical log might be too small to accommodate the time it takes to flush the buffer pool during checkpoint processing. The server might block transactions during checkpoints.
If the server blocks transactions, increase the physical log size to at least 11391264 KB.

Index builds

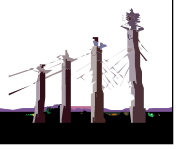
- How is an index built
- How to improve performance?
  - Smaller indexes
    - DS_NONPDQ_MEMORY
  - Bigger indexes
    - Tweak sort parameters



How is an index built?

- Read through the data
  - Either single or fragmented
- Sort the data
  - Got either one set or a number of sets
  - Merge the data, if necessary

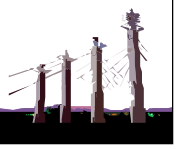
Do as much in memory as possible.



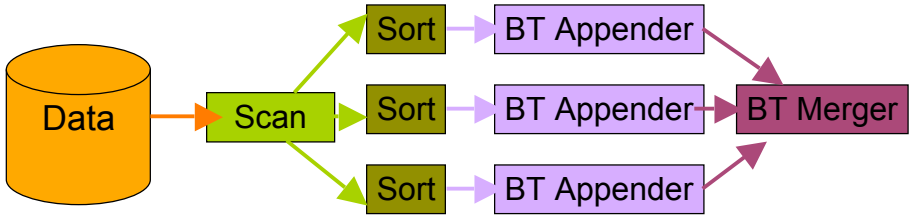
Index build: DS_NONPDQ_MEMORY

- Tune DS_NONPDQ_QUERY_MEM
  - memory given to sorts without PDQ
  - Minimum Value=Default 128KB
  - Maximum Value 25% of DS_TOTAL MEMORY
  - Change while instance is online

    onmode -wf DS_NONPDQ_QUERY_MEM=5000



How is an index built (using PDQ)?



Rule of thumb: Sorts in memory are good, sort on disks are slower.

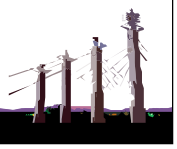
So make sure you can sort in memory.

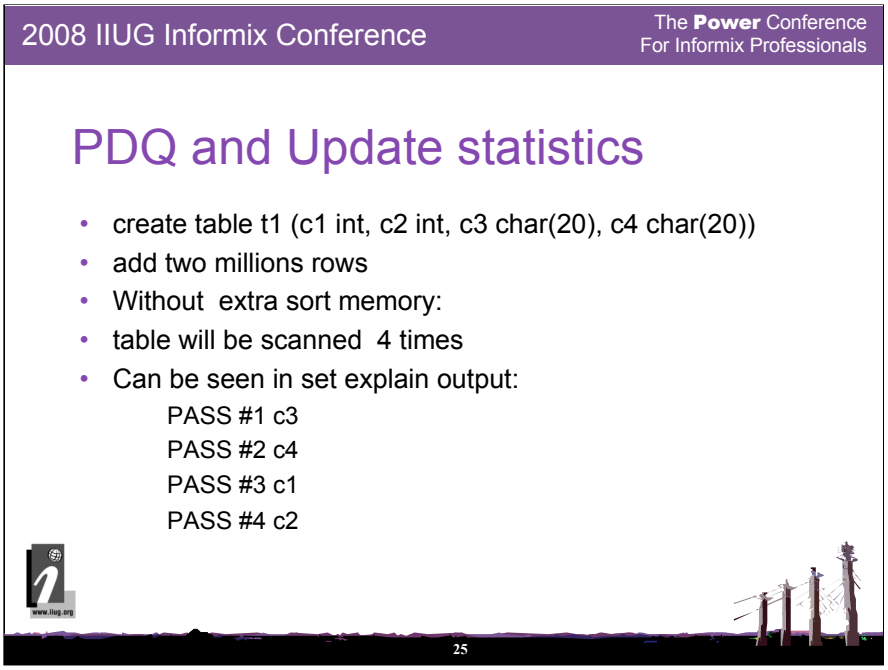
Also the more sort threads we have the more merges need to be done.

So there might be an optimization issue here, regarding the number of sort theads.

Large Index builds

- Ideal: sort in memory
- Enable PDQ
  - PDQPRIORITY
  - DS_TOTAL_MEMORY
  - DS_MAX_QUERIES
- PSORT_NPROCS





Here are the results for three different runs of update statistics:

UPDATE STATISTICS:

==========

Table: ebach.t1

Mode: HIGH

Number of Bins: 267 Bin size 10813

Sort data 166.1 MB Sort memory granted 15.0 MB

Estimated number of table scans 4

PASS #1 c3

PASS #2 c4

PASS #3 c1

PASS #4 c2

Scan 7 Sort 1 Build 1 Insert 0 Close 0 Total 9

Completed pass 1 in 0 minutes 9 seconds

Scan 7 Sort 0 Build 2 Insert 0 Close 0 Total 9

Completed pass 2 in 0 minutes 9 seconds

Scan 5 Sort 1 Build 0 Insert 0 Close 0 Total 6

Completed pass 3 in 0 minutes 6 seconds

Set PDQ PRIO 25

UPDATE STATISTICS:

==========

Table: ebach.t1

PDQ with Update Statistics

- PDQ can change the number of table scans needed.
- Use set explain

```
Table:          ebach.t1
Mode:           HIGH
Number of Bins:     267        Bin size
    10813
Sort data          166.1 MB
PDQ memory granted         174.4 MB
Estimated number of table scans 1
PASS #1 c1,c2,c3,c4
Light scans enabled
Scan 4 Sort 8 Build 3 Insert 0 Close 0 Total
    15
Completed pass 1 in 0 minutes 15 seconds
```

PDQ Memory

Features Enabled

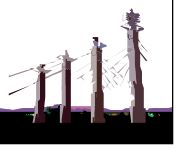
Total Time needed

A few more things

- External directives
- Fragmentation guidelines
- Avoid round robin fragmentation
- Check for limitations on data types esp. varchar

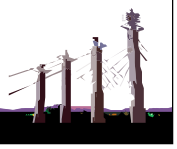
Summary

- Get a baseline of your system
  - Performance
  - SQLs
- Stick to your processes
  - Get verification for assumptions
- A few helpful parameters
  - DONTDRAINPOOLS
  - DS_NONPDQ_MEMORY
  - PSORT_NPROCS



More Info

- An article stating where to find information:
  http://www-1.ibm.com/support/docview.wss?uid=swg21250290
- Documentation
  http://publib.boulder.ibm.com/infocenter/idshelp/v111/index.jsp?topic=/com.ibm.perf.doc/perf.htm
- DCF articles and white papers
- http://www-306.ibm.com/software/sw-library/en_US/detail/F415342F55896R80.html



Q&A

THANK YOU

Session D01

Top Tips learned from Performance PMRS

Dr. Elisabeth Bach

IBM

elisabeth.bach@de.ibm.com