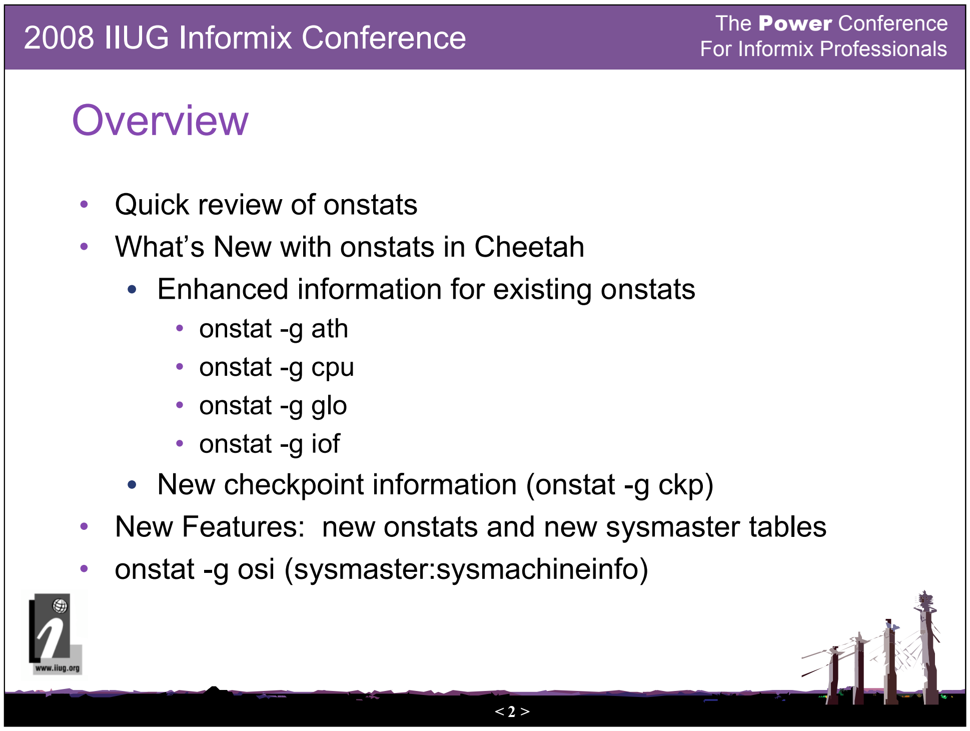
Understanding New onstat Options in Cheetah

Hyun-Ju Vega
IBM

Session 121 (Alternate)
Monday, April 00, 2008 • 00:00 a.m. – 00:00 a.m.

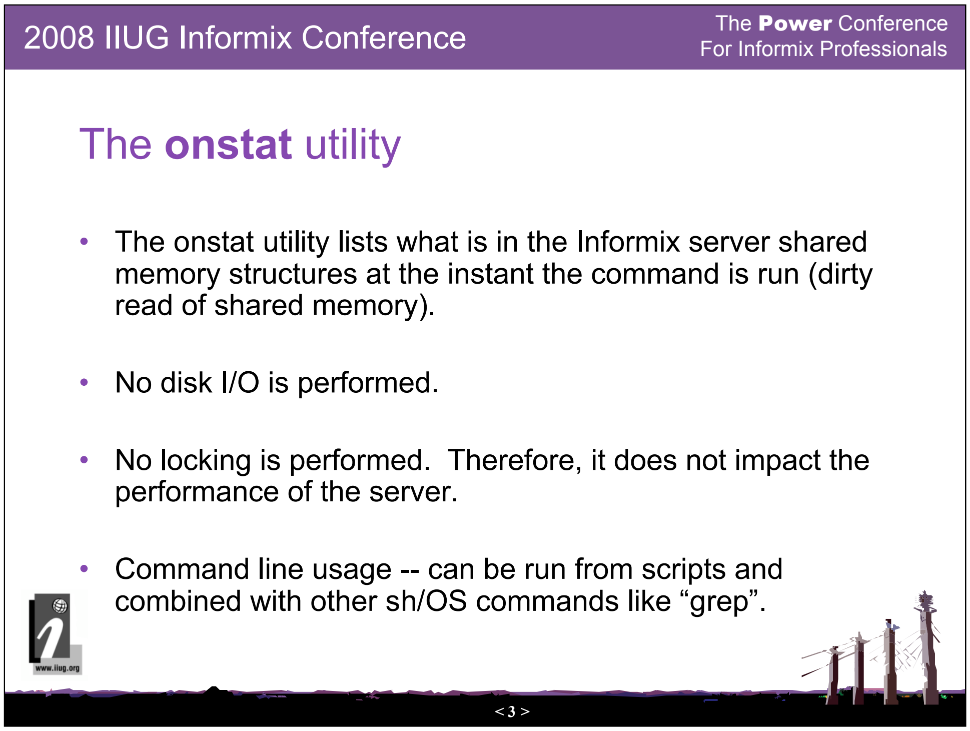
2008 IIUG Informix Conference

www.iiug.org



Overview

- Quick review of onstats
- What's New with onstats in Cheetah
  - Enhanced information for existing onstats
    - onstat -g ath
    - onstat -g cpu
    - onstat -g glo
    - onstat -g iof
  - New checkpoint information (onstat -g ckp)
- New Features: new onstats and new sysmaster tables
- onstat -g osi (sysmaster:sysmachineinfo)



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

The **onstat** utility

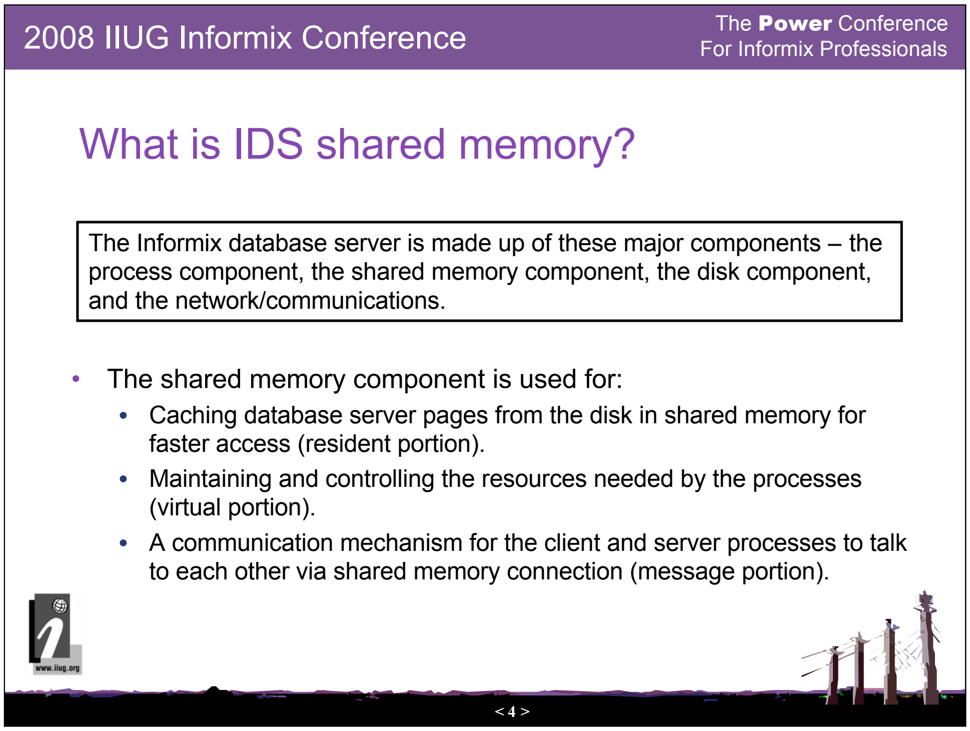
- The onstat utility lists what is in the Informix server shared memory structures at the instant the command is run (dirty read of shared memory).
- No disk I/O is performed.
- No locking is performed. Therefore, it does not impact the performance of the server.
- Command line usage -- can be run from scripts and combined with other sh/OS commands like “grep”.

< 3 >

The **onstat** utility reads the server shared-memory structures and reports the contents of shared memory at the *instant* that it is run. This means the contents of shared memory might be changing as **onstat** results are printed (as no memory locking is done by **onstat**).

The **onstat** utility prints out the contents of the various internal tables (or data structures) maintained in shared memory. Since the data structures and internal tables reflect the current state of the server, this report gives a good snapshot of what’s going on in the server.

Generally, **onstat** does no disk I/O; it reads from shared memory alone (there are a few options that read from disk files). Because it places no locks on shared memory resources, it does not impact the performance of the server.



2008 IIUG Informix Conference

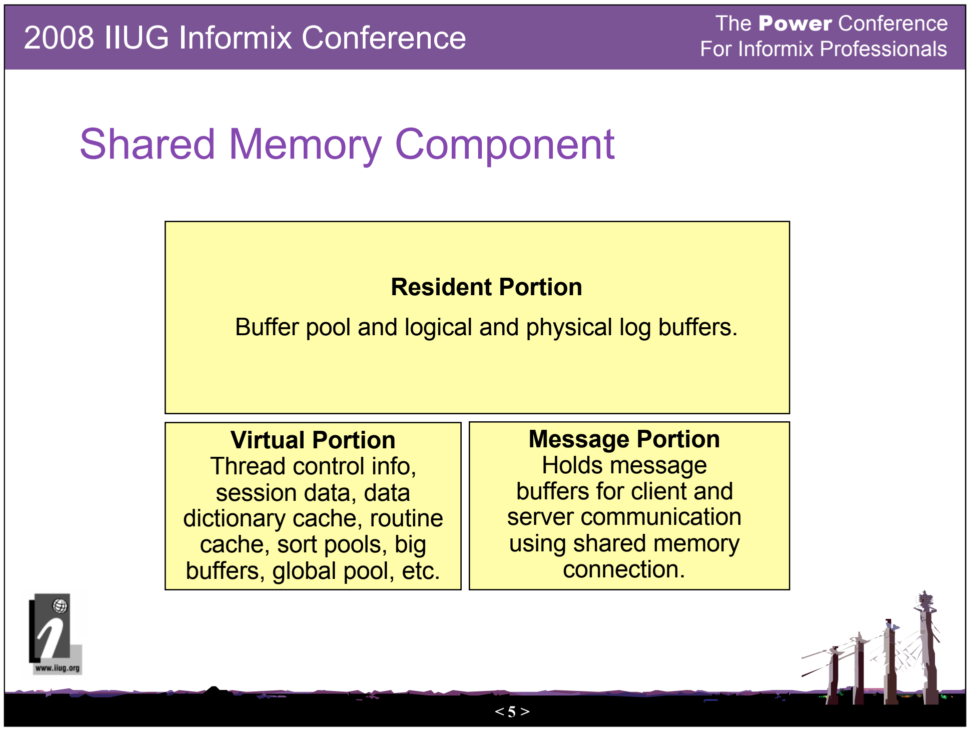
The **Power** Conference
For Informix Professionals

What is IDS shared memory?

The Informix database server is made up of these major components – the process component, the shared memory component, the disk component, and the network/communications.

- The shared memory component is used for:
  - Caching database server pages from the disk in shared memory for faster access (resident portion).
  - Maintaining and controlling the resources needed by the processes (virtual portion).
  - A communication mechanism for the client and server processes to talk to each other via shared memory connection (message portion).

onstat -g seg displays the IDS shared memory segments



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

Shared Memory Component

Resident Portion

Buffer pool and logical and physical log buffers.

Virtual Portion
Thread control info, session data, data dictionary cache, routine cache, sort pools, big buffers, global pool, etc.

Message Portion
Holds message buffers for client and server communication using shared memory connection.

The size of the resident portion is determined by the ONCONFIG parameter BUFFERS/BUFFERPOOL PHYSBUFF, LOGBUFF.

The resident portion of the shared memory can be configured to remain resident in main memory (if the OS supports this capability).

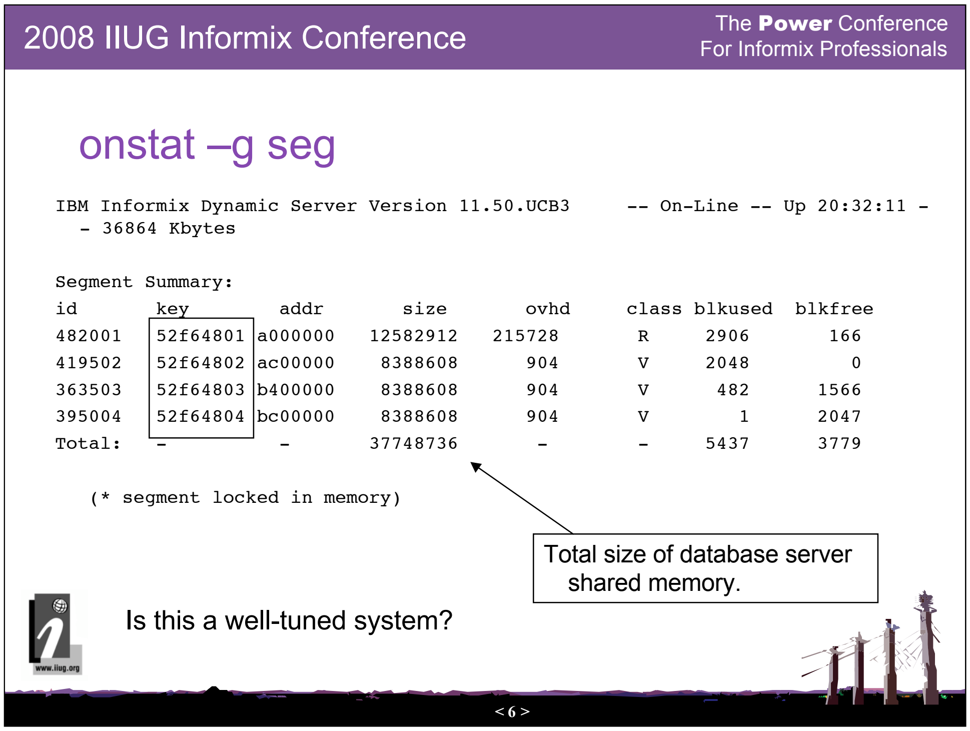
The initial size of the virtual portion is determined by the ONCONFIG parameter SHMVIRTSIZE.

Subsequent size of the virtual portion is determined by the ONCONFIG parameter SHMADD.

The size of the virtual portion can grow and shrink dynamically.

The size of the message portion is determined by the NETTYPE configuration parameter – the below NETTYPE configuration shows shared memory communication protocol, number of poll threads, number of users per poll thread, and the virtual processor (vp) class on which to run the poll threads. How many connections are configured in total?

```
NETTYPE    ipcshm,4,50,CPU
```



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

onstat –g seg

```
IBM Informix Dynamic Server Version 11.50.UCB3     -- On-Line -- Up 20:32:11 -
  - 36864 Kbytes

Segment Summary:
id         key        addr       size       ovhd      class blkused  blkfree
482001    52f64801 a000000   12582912   215728      R      2906       166
419502    52f64802 ac00000    8388608      904      V      2048         0
363503    52f64803 b400000    8388608      904      V       482      1566
395004    52f64804 bc00000    8388608      904      V         1      2047
Total:    -           -      37748736       -       -      5437      3779

   (* segment locked in memory)
```

Total size of database server shared memory.

Is this a well-tuned system?

< 6 >

The **onstat -g seg** above shows that the total size of the database server shared memory is 37748736 bytes.

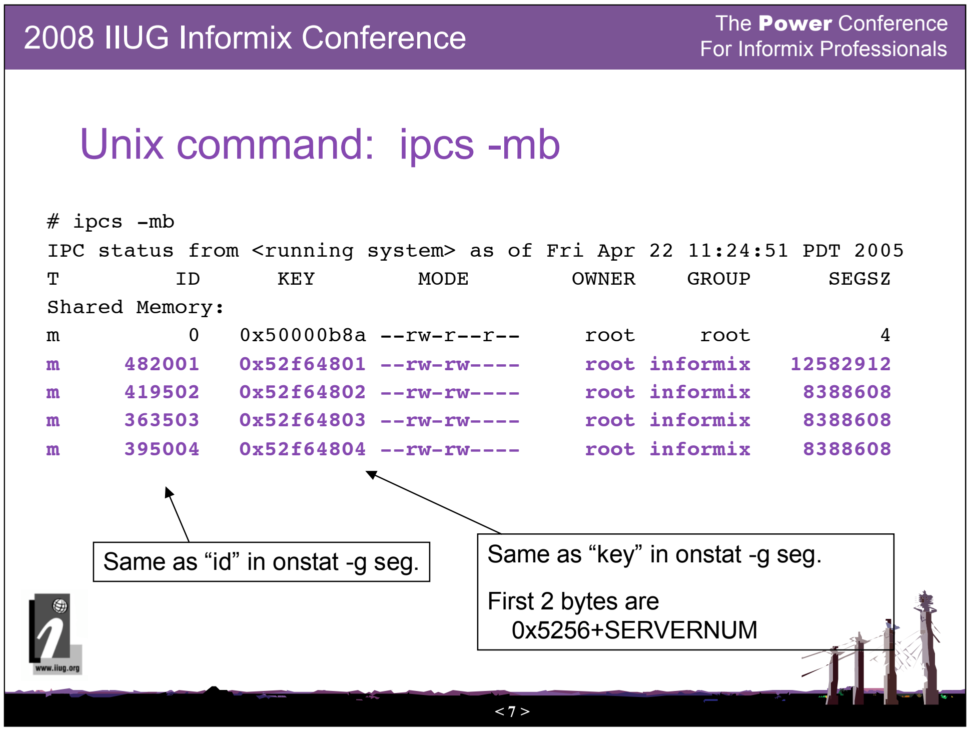
Starting in 11.50, the key is displayed in hex (instead of decimal) since the ipcs output displays the info in hex.

```
# ipcs -mb
IPC status from <running system> as of Fri Apr 22 11:24:51 PDT 2005
T       ID     KEY        MODE        OWNER   GROUP     SEGSZ
Shared Memory:
m            0   0x50000b8a --rw-r--r--     root     root         4
m     482001   0x52f64801 --rw-rw----     root informix   12582912
m     419502   0x52f64802 --rw-rw----     root informix    8388608
m     363503   0x52f64803 --rw-rw----     root informix    8388608
m     395004   0x52f64804 --rw-rw----     root informix    8388608.
```

SERVERNUM 160 (0xa0) ==> 0x5256 + 0xa0 = 0x52f6

SHMVIRTSIZE 8000
SHMADD 8192

How can the configuration be improved?
Why is there no message segment?



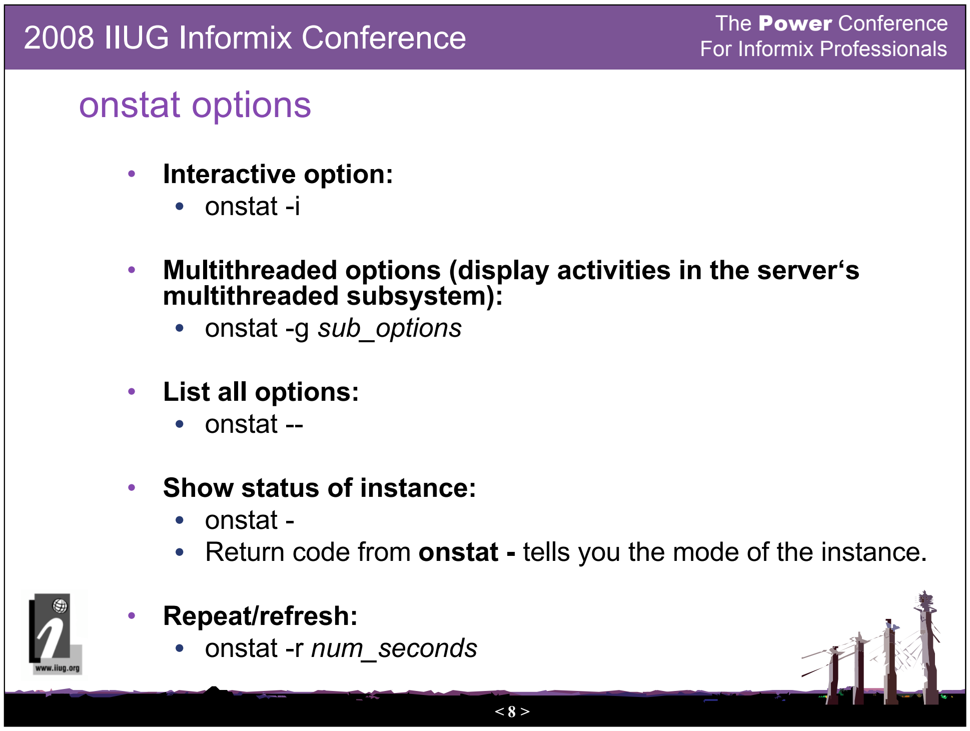
Unix command: ipcs -mb

```
# ipcs -mb
IPC status from <running system> as of Fri Apr 22 11:24:51 PDT 2005
T         ID      KEY         MODE        OWNER    GROUP      SEGSZ
Shared Memory:
m          0    0x50000b8a --rw-r--r--      root     root          4
m      482001   0x52f64801 --rw-rw----      root informix   12582912
m      419502   0x52f64802 --rw-rw----      root informix    8388608
m      363503   0x52f64803 --rw-rw----      root informix    8388608
m      395004   0x52f64804 --rw-rw----      root informix    8388608
```

Same as “id” in onstat -g seg.

Same as “key” in onstat -g seg.

First 2 bytes are
0x5256+SERVERNUM



onstat options

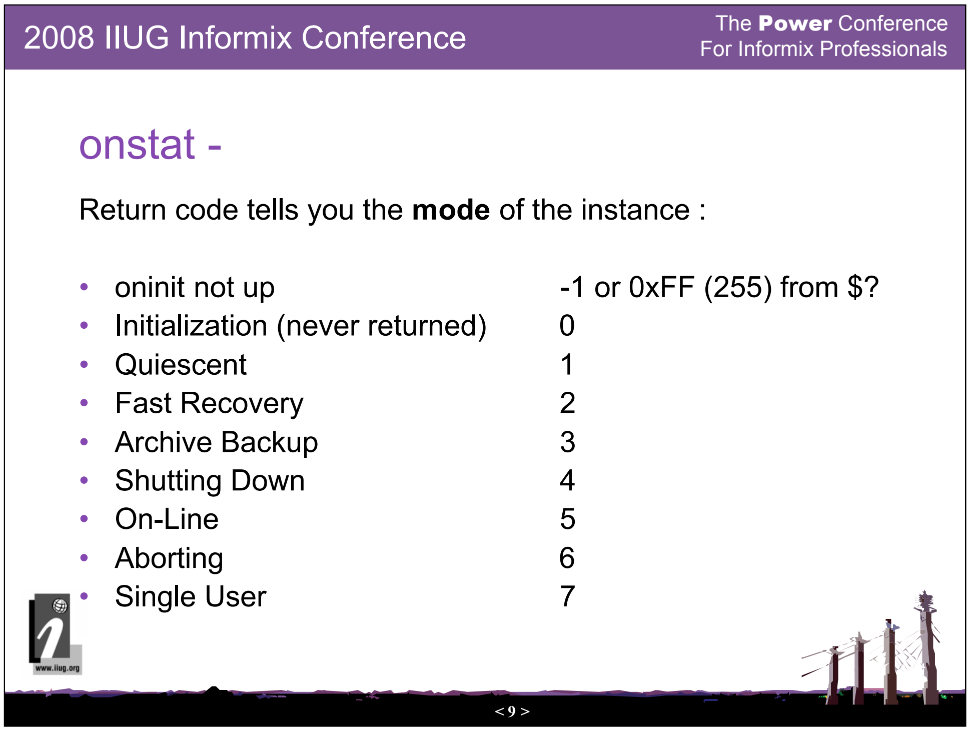
- **Interactive option:**
  - onstat -i

- **Multithreaded options (display activities in the server‘s multithreaded subsystem):**
  - onstat -g *sub_options*

- **List all options:**
  - onstat --

- **Show status of instance:**
  - onstat -
  - Return code from **onstat -** tells you the mode of the instance.

- **Repeat/refresh:**
  - onstat -r *num_seconds*

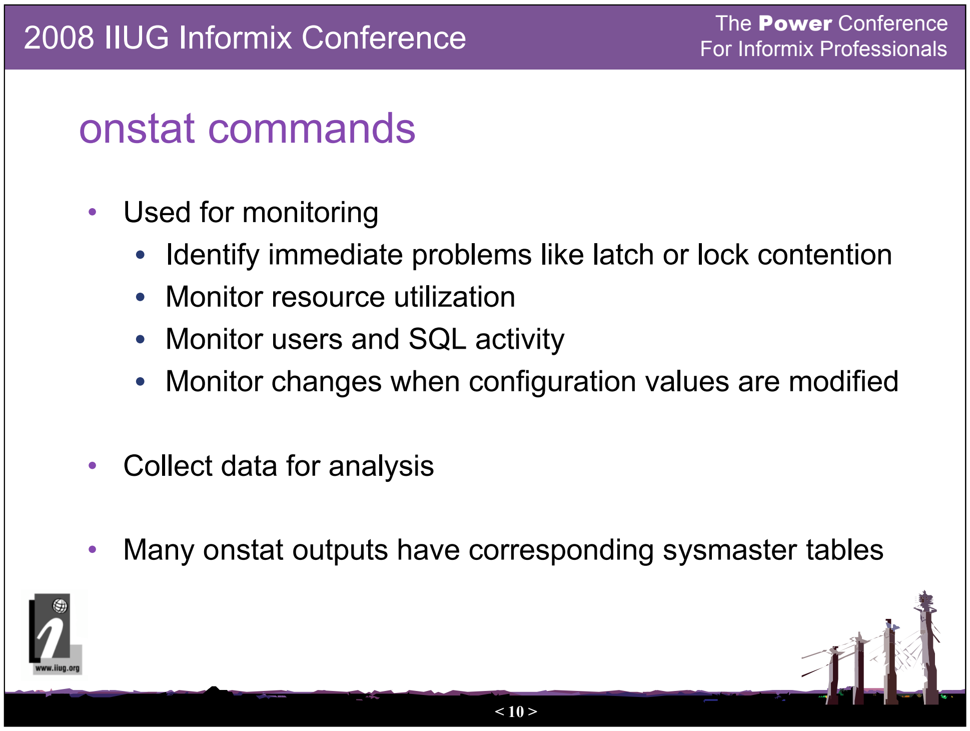


onstat -

Return code tells you the **mode** of the instance :

oninit not up	-1 or 0xFF (255) from $?
Initialization (never returned)	0
Quiescent	1
Fast Recovery	2
Archive Backup	3
Shutting Down	4
On-Line	5
Aborting	6
Single User	7

www.iiug.org

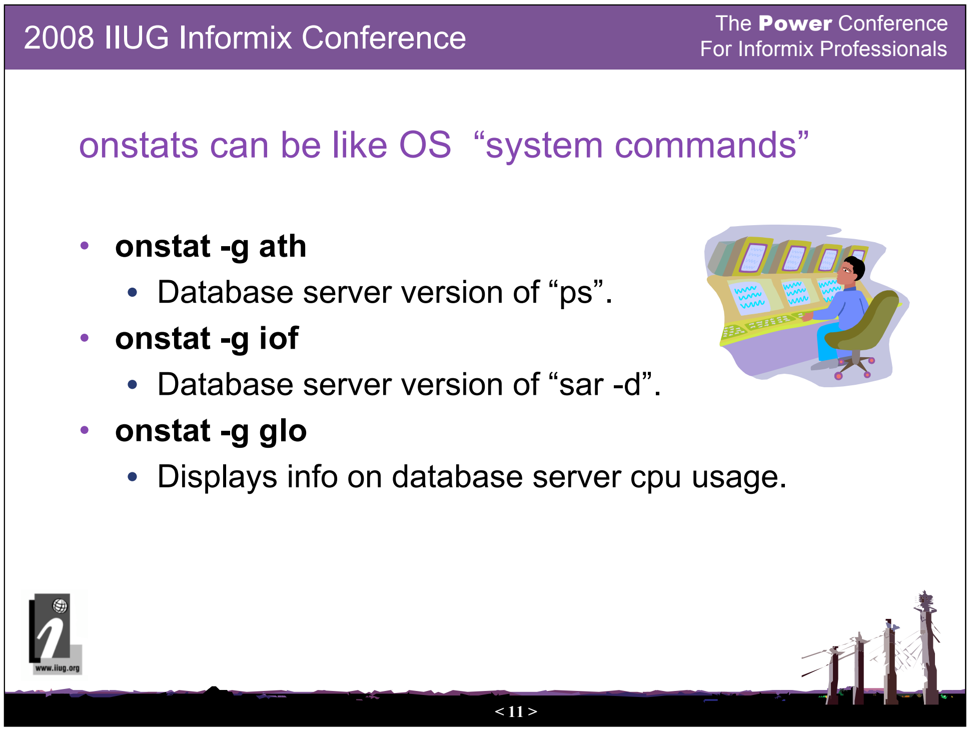


onstat commands

- Used for monitoring
  - Identify immediate problems like latch or lock contention
  - Monitor resource utilization
  - Monitor users and SQL activity
  - Monitor changes when configuration values are modified

- Collect data for analysis

- Many onstat outputs have corresponding sysmaster tables



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

onstats can be like OS "system commands"

- **onstat -g ath**
  - Database server version of "ps".
- **onstat -g iof**
  - Database server version of "sar -d".
- **onstat -g glo**
  - Displays info on database server cpu usage.

< 11 >

onstat -g ath :

Provides a quick check for what threads are currently active in the system.

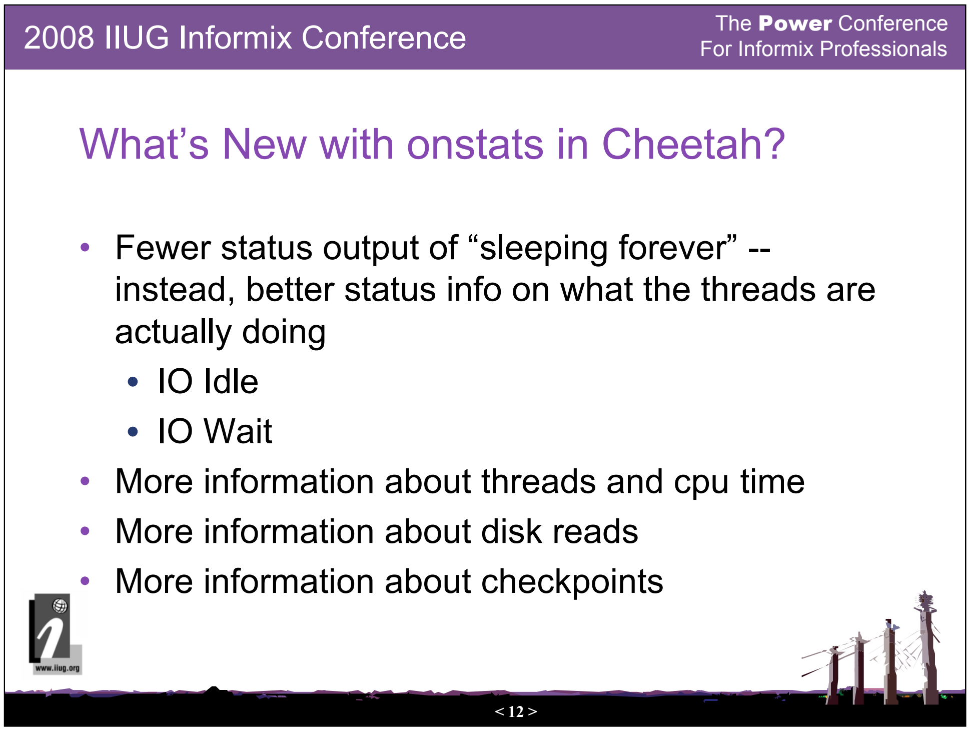
A glance through the output will quickly reveal whether KAIO threads are running, if there are any sessions running via sqlexec threads, or if there are a large number of scan and exchange threads running to handle parallel operations.

onstat -g iof :

Displays the statistics for disk I/O by chunk/file. Examination of the values in each of the operations columns can identify heavy I/Os against a particular device or chunk.

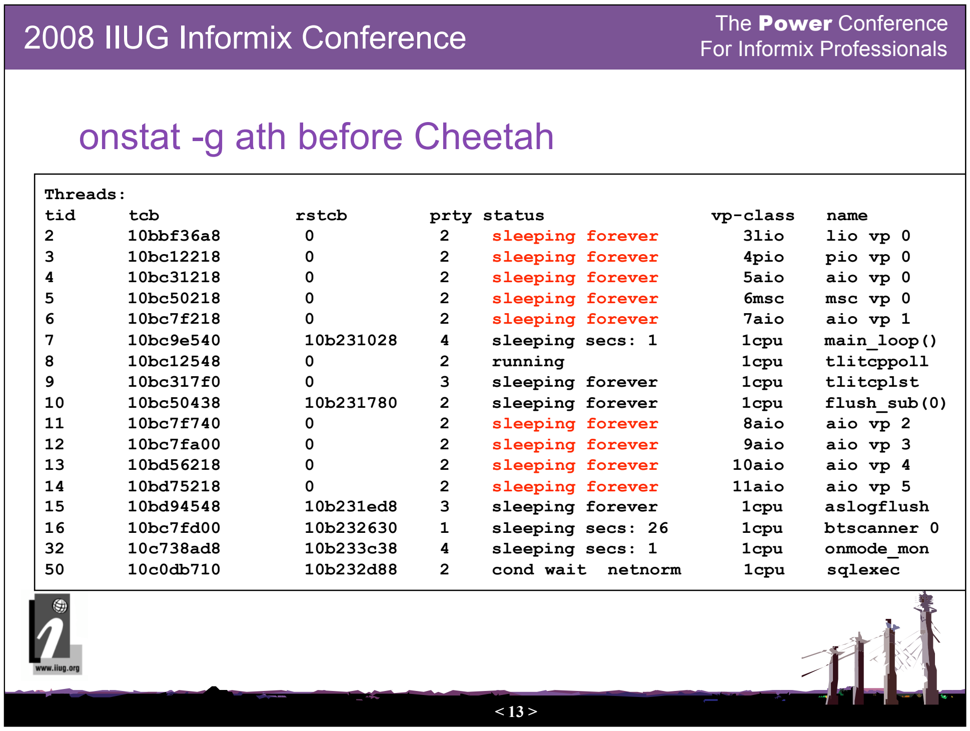
onstat -g glo:

Listing the database server virtual processors, this onstat display information on cpu usage. In 11.50, onstat -g glo output was enhanced to show the percentage of time that the threads are actually running on a processor when the thread status is shown as "running".



What’s New with onstats in Cheetah?

- Fewer status output of “sleeping forever” -- instead, better status info on what the threads are actually doing
  - IO Idle
  - IO Wait
- More information about threads and cpu time
- More information about disk reads
- More information about checkpoints



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

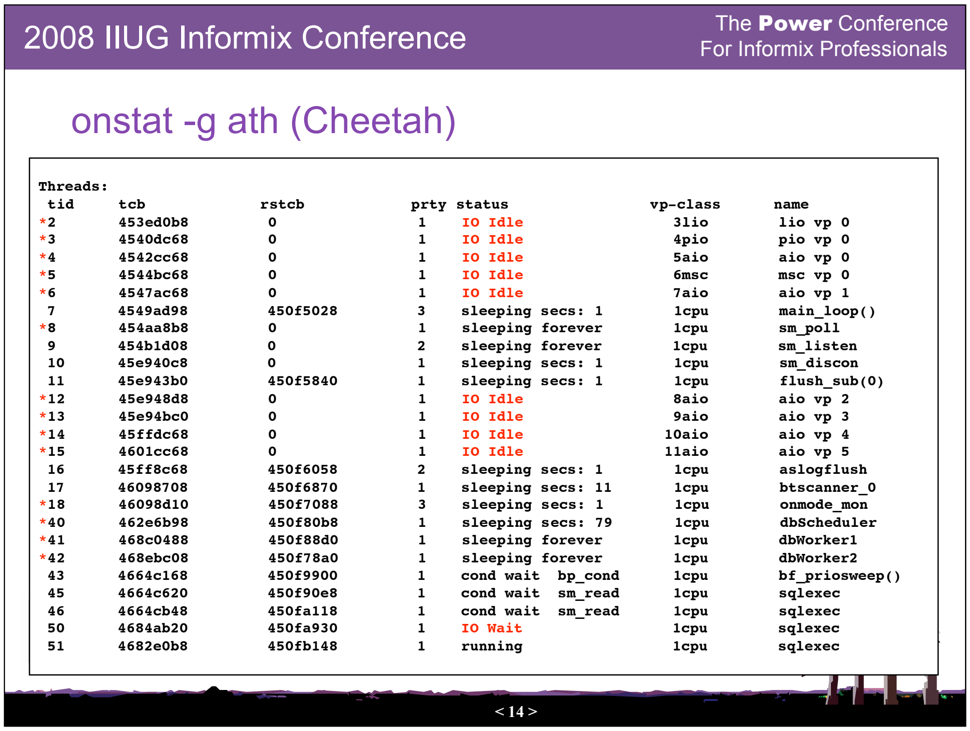
onstat -g ath before Cheetah

```
Threads:
tid      tcb              rstcb         prty status                vp-class    name
2        10bbf36a8         0             2    sleeping forever        3lio     lio vp 0
3        10bc12218         0             2    sleeping forever        4pio     pio vp 0
4        10bc31218         0             2    sleeping forever        5aio     aio vp 0
5        10bc50218         0             2    sleeping forever        6msc     msc vp 0
6        10bc7f218         0             2    sleeping forever        7aio     aio vp 1
7        10bc9e540         10b231028     4    sleeping secs: 1        1cpu     main_loop()
8        10bc12548         0             2    running                 1cpu     tlitcppoll
9        10bc317f0         0             3    sleeping forever        1cpu     tlitcplst
10       10bc50438         10b231780     2    sleeping forever        1cpu     flush_sub(0)
11       10bc7f740         0             2    sleeping forever        8aio     aio vp 2
12       10bc7fa00         0             2    sleeping forever        9aio     aio vp 3
13       10bd56218         0             2    sleeping forever       10aio     aio vp 4
14       10bd75218         0             2    sleeping forever       11aio     aio vp 5
15       10bd94548         10b231ed8     3    sleeping forever        1cpu     aslogflush
16       10bc7fd00         10b232630     1    sleeping secs: 26       1cpu     btscanner 0
32       10c738ad8         10b233c38     4    sleeping secs: 1        1cpu     onmode_mon
50       10c0db710         10b232d88     2    cond wait   netnorm     1cpu     sqlexec
```

< 13 >

In many cases the server indicates that threads are waiting forever, but we don’t know what they’re waiting for.

For example, if several threads are waiting for I/O operations to complete, the thread status is “sleeping forever”, but in reality, they are sleeping until an IO operation completes.



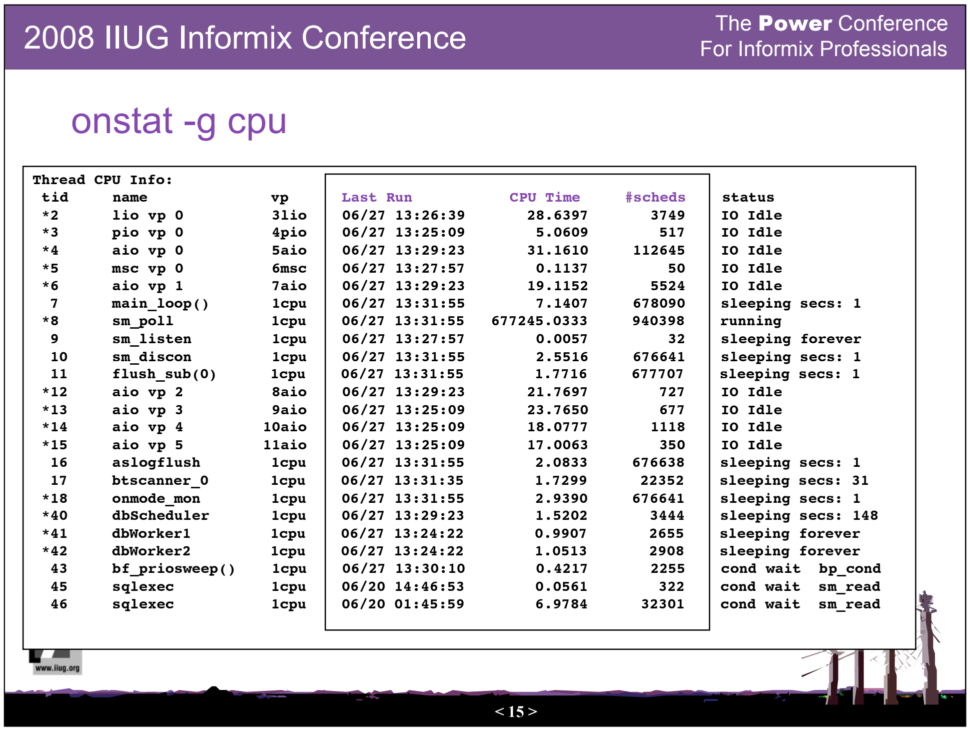
2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

onstat -g ath (Cheetah)

```
Threads:
 tid     tcb             rstcb          prty status                 vp-class      name
*2       453ed0b8         0              1    IO Idle                 3lio        lio vp 0
*3       4540dc68         0              1    IO Idle                 4pio        pio vp 0
*4       4542cc68         0              1    IO Idle                 5aio        aio vp 0
*5       4544bc68         0              1    IO Idle                 6msc        msc vp 0
*6       4547ac68         0              1    IO Idle                 7aio        aio vp 1
 7       4549ad98         450f5028       3    sleeping secs: 1        1cpu        main_loop()
*8       454aa8b8         0              1    sleeping forever        1cpu        sm_poll
 9       454b1d08         0              2    sleeping forever        1cpu        sm_listen
 10      45e940c8         0              1    sleeping secs: 1        1cpu        sm_discon
 11      45e943b0         450f5840       1    sleeping secs: 1        1cpu        flush_sub(0)
*12      45e948d8         0              1    IO Idle                 8aio        aio vp 2
*13      45e94bc0         0              1    IO Idle                 9aio        aio vp 3
*14      45ffdc68         0              1    IO Idle                10aio        aio vp 4
*15      4601cc68         0              1    IO Idle                11aio        aio vp 5
 16      45ff8c68         450f6058       2    sleeping secs: 1        1cpu        aslogflush
 17      46098708         450f6870       1    sleeping secs: 11       1cpu        btscanner_0
*18      46098d10         450f7088       3    sleeping secs: 1        1cpu        onmode_mon
*40      462e6b98         450f80b8       1    sleeping secs: 79       1cpu        dbScheduler
*41      468c0488         450f88d0       1    sleeping forever        1cpu        dbWorker1
*42      468ebc08         450f78a0       1    sleeping forever        1cpu        dbWorker2
 43      4664c168         450f9900       1    cond wait  bp_cond      1cpu        bf_priosweep()
 45      4664c620         450f90e8       1    cond wait  sm_read      1cpu        sqlexec
 46      4664cb48         450fa118       1    cond wait  sm_read      1cpu        sqlexec
 50      4684ab20         450fa930       1    IO Wait                 1cpu        sqlexec
 51      4682e0b8         450fb148       1    running                 1cpu        sqlexec
```

* means that the thread is bound on the vp



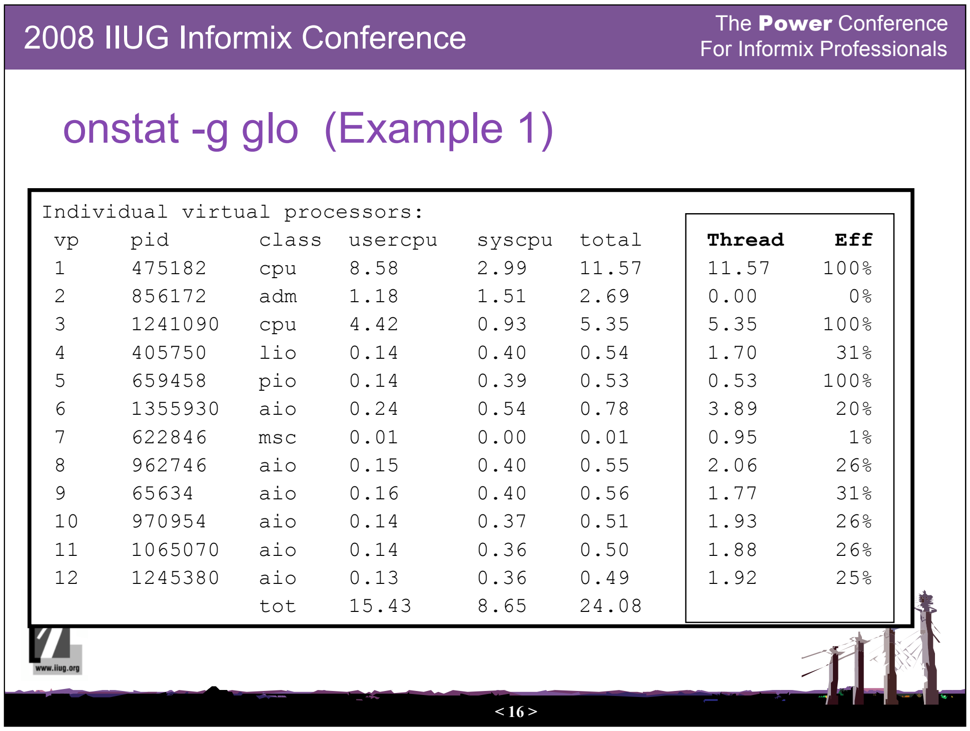
2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

onstat -g cpu

```
Thread CPU Info:
 tid      name               vp     Last Run          CPU Time    #scheds    status
 *2       lio vp 0           3lio   06/27 13:26:39     28.6397       3749    IO Idle
 *3       pio vp 0           4pio   06/27 13:25:09      5.0609        517    IO Idle
 *4       aio vp 0           5aio   06/27 13:29:23     31.1610     112645    IO Idle
 *5       msc vp 0           6msc   06/27 13:27:57      0.1137         50    IO Idle
 *6       aio vp 1           7aio   06/27 13:29:23     19.1152       5524    IO Idle
  7       main_loop()        1cpu   06/27 13:31:55      7.1407     678090    sleeping secs: 1
 *8       sm_poll            1cpu   06/27 13:31:55  677245.0333    940398    running
  9       sm_listen          1cpu   06/27 13:27:57      0.0057         32    sleeping forever
  10      sm_discon          1cpu   06/27 13:31:55      2.5516     676641    sleeping secs: 1
  11      flush_sub(0)       1cpu   06/27 13:31:55      1.7716     677707    sleeping secs: 1
 *12      aio vp 2           8aio   06/27 13:29:23     21.7697        727    IO Idle
 *13      aio vp 3           9aio   06/27 13:25:09     23.7650        677    IO Idle
 *14      aio vp 4          10aio   06/27 13:25:09     18.0777       1118    IO Idle
 *15      aio vp 5          11aio   06/27 13:25:09     17.0063        350    IO Idle
  16      aslogflush         1cpu   06/27 13:31:55      2.0833     676638    sleeping secs: 1
  17      btscanner_0        1cpu   06/27 13:31:35      1.7299      22352    sleeping secs: 31
 *18      onmode_mon         1cpu   06/27 13:31:55      2.9390     676641    sleeping secs: 1
 *40      dbScheduler        1cpu   06/27 13:29:23      1.5202       3444    sleeping secs: 148
 *41      dbWorker1          1cpu   06/27 13:24:22      0.9907       2655    sleeping forever
 *42      dbWorker2          1cpu   06/27 13:24:22      1.0513       2908    sleeping forever
  43      bf_priosweep()     1cpu   06/27 13:30:10      0.4217       2255    cond wait  bp_cond
  45      sqlexec            1cpu   06/20 14:46:53      0.0561        322    cond wait  sm_read
  46      sqlexec            1cpu   06/20 01:45:59      6.9784      32301    cond wait  sm_read
```

onstat -g cpu displays information about how much cpu time each thread has incurred.



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

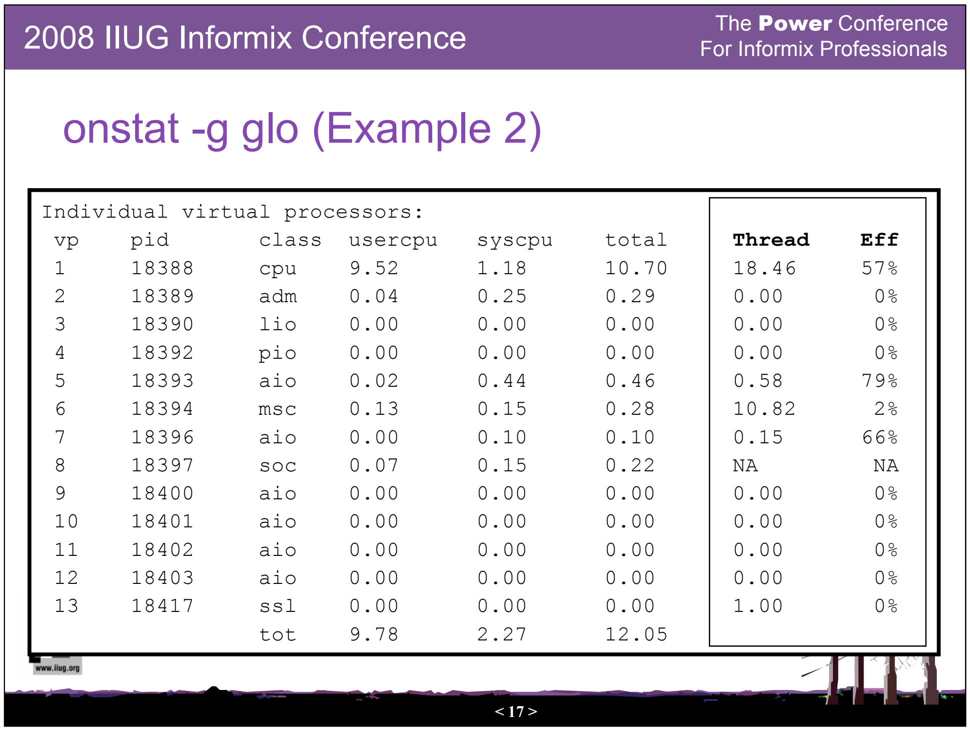
onstat -g glo (Example 1)

```
Individual virtual processors:
 vp     pid       class  usercpu    syscpu   total      Thread     Eff
 1      475182    cpu    8.58       2.99     11.57      11.57      100%
 2      856172    adm    1.18       1.51     2.69       0.00         0%
 3      1241090   cpu    4.42       0.93     5.35       5.35       100%
 4      405750    lio    0.14       0.40     0.54       1.70        31%
 5      659458    pio    0.14       0.39     0.53       0.53       100%
 6      1355930   aio    0.24       0.54     0.78       3.89        20%
 7      622846    msc    0.01       0.00     0.01       0.95         1%
 8      962746    aio    0.15       0.40     0.55       2.06        26%
 9      65634     aio    0.16       0.40     0.56       1.77        31%
 10     970954    aio    0.14       0.37     0.51       1.93        26%
 11     1065070   aio    0.14       0.36     0.50       1.88        26%
 12     1245380   aio    0.13       0.36     0.49       1.92        25%
                  tot    15.43      8.65     24.08
```

www.iiug.org

onstat -g glo has been enhanced to show the percentage of time that the threads are actually running on a processor when the thread status is shown as “running”. The thread numbers do not include the times spent polling.

These numbers are more reflective of what's going on in the system during steady state (not during startup).



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

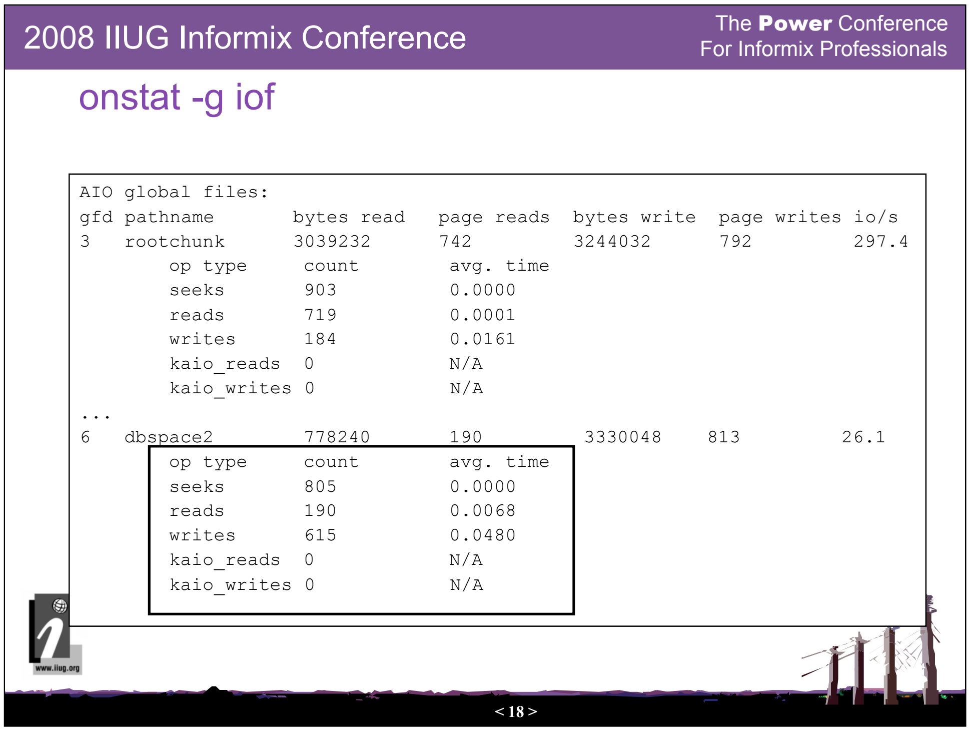
onstat -g glo (Example 2)

```
Individual virtual processors:
 vp     pid        class   usercpu    syscpu     total       Thread     Eff
 1      18388      cpu     9.52       1.18       10.70       18.46      57%
 2      18389      adm     0.04       0.25       0.29        0.00        0%
 3      18390      lio     0.00       0.00       0.00        0.00        0%
 4      18392      pio     0.00       0.00       0.00        0.00        0%
 5      18393      aio     0.02       0.44       0.46        0.58       79%
 6      18394      msc     0.13       0.15       0.28        10.82       2%
 7      18396      aio     0.00       0.10       0.10        0.15       66%
 8      18397      soc     0.07       0.15       0.22        NA          NA
 9      18400      aio     0.00       0.00       0.00        0.00        0%
 10     18401      aio     0.00       0.00       0.00        0.00        0%
 11     18402      aio     0.00       0.00       0.00        0.00        0%
 12     18403      aio     0.00       0.00       0.00        0.00        0%
 13     18417      ssl     0.00       0.00       0.00        1.00        0%
                   tot     9.78       2.27       12.05
```

www.iiug.org

< 17 >

In this example, for vp1, the threads ran for 18.5 seconds, but the process 18388 actually ran on a physical processor for only 10.7 seconds -- so, part of the time (43% of the time) that the IDS info shows the threads as "running“ on cpu vp1, the threads are not really doing anything since the vp itself is not running on a processor (the OS isn't letting the cpu vp process run on the processor). That's where the Eff number comes from ... you can interpret that as 57% of the time the sqlexec or daemon (flusher, btree, admin threads, etc.) threads are shown as "running", they're not actually executing code on a processor -- IDS shows them as running on the cpu vp, but the cpu vp may not be running on a processor.



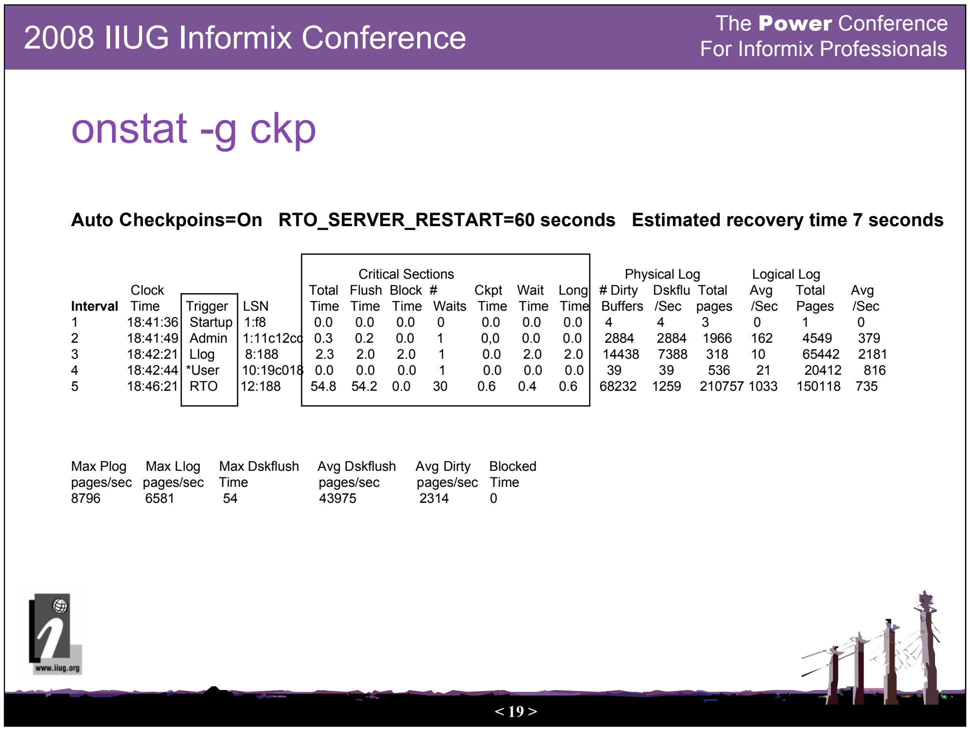
2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

onstat -g iof

```
AIO global files:
gfd pathname       bytes read   page reads  bytes write  page writes io/s
3   rootchunk      3039232      742         3244032      792         297.4
        op type     count        avg. time
        seeks       903          0.0000
        reads       719          0.0001
        writes      184          0.0161
        kaio_reads  0            N/A
        kaio_writes 0            N/A
...
6   dbspace2        778240       190          3330048    813         26.1
        op type     count        avg. time
        seeks       805          0.0000
        reads       190          0.0068
        writes      615          0.0480
        kaio_reads  0            N/A
        kaio_writes 0            N/A
```

In IDS 11, onstat -g iof has been enhanced to show the different types of I/O operations on a chunk or device.



2008 IIUG Informix Conference

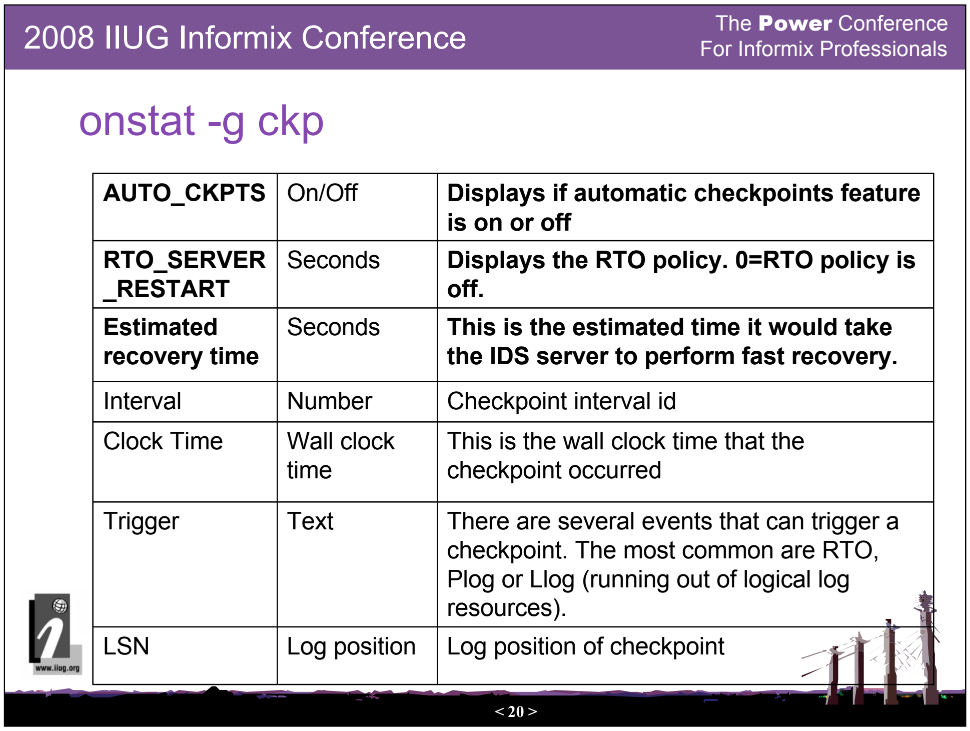
The **Power** Conference
For Informix Professionals

onstat -g ckp

```
Auto Checkpoins=On   RTO_SERVER_RESTART=60 seconds   Estimated recovery time 7 seconds

                                         Critical Sections                          Physical Log          Logical Log
          Clock                          Total  Flush  Block  #      Ckpt  Wait  Long  # Dirty  Dskflu  Total   Avg    Total   Avg
Interval  Time      Trigger  LSN         Time   Time   Time   Waits  Time  Time  Time  Buffers  /Sec    pages   /Sec   Pages   /Sec
1         18:41:36  Startup  1:f8        0.0    0.0    0.0    0      0.0   0.0   0.0   4        4       3       0      1       0
2         18:41:49  Admin    1:11c12cc   0.3    0.2    0.0    1      0,0   0.0   0.0   2884     2884    1966    162    4549    379
3         18:42:21  Llog     8:188       2.3    2.0    2.0    1      0.0   2.0   2.0   14438    7388    318     10     65442   2181
4         18:42:44  *User    10:19c018   0.0    0.0    0.0    1      0.0   0.0   0.0   39       39      536     21     20412   816
5         18:46:21  RTO      12:188      54.8   54.2   0.0    30     0.6   0.4   0.6   68232    1259    210757  1033   150118  735


Max Plog   Max Llog   Max Dskflush   Avg Dskflush   Avg Dirty   Blocked
pages/sec  pages/sec  Time           pages/sec      pages/sec   Time
8796       6581       54             43975          2314        0
```

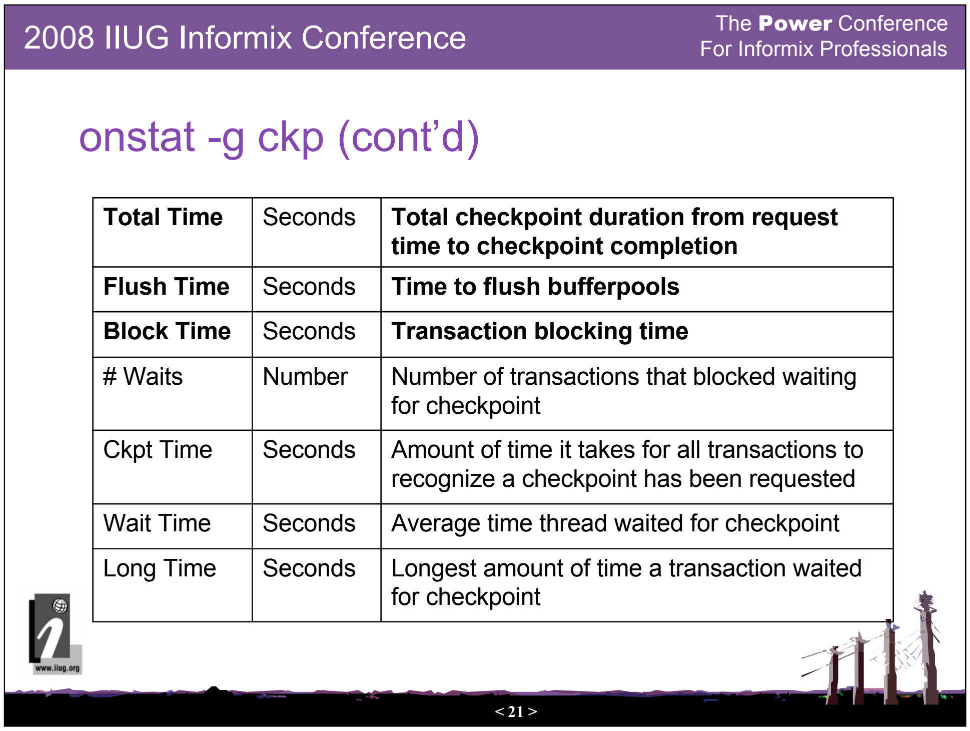
www.iiug.org

In IDS 11, a lot more information may be obtained about each checkpoint.



onstat -g ckp

AUTO_CKPTS	On/Off	**Displays if automatic checkpoints feature is on or off**
RTO_SERVER_RESTART	Seconds	**Displays the RTO policy. 0=RTO policy is off.**
Estimated recovery time	Seconds	**This is the estimated time it would take the IDS server to perform fast recovery.**
Interval	Number	Checkpoint interval id
Clock Time	Wall clock time	This is the wall clock time that the checkpoint occurred
Trigger	Text	There are several events that can trigger a checkpoint. The most common are RTO, Plog or Llog (running out of logical log resources).
LSN	Log position	Log position of checkpoint



2008 IIUG Informix Conference

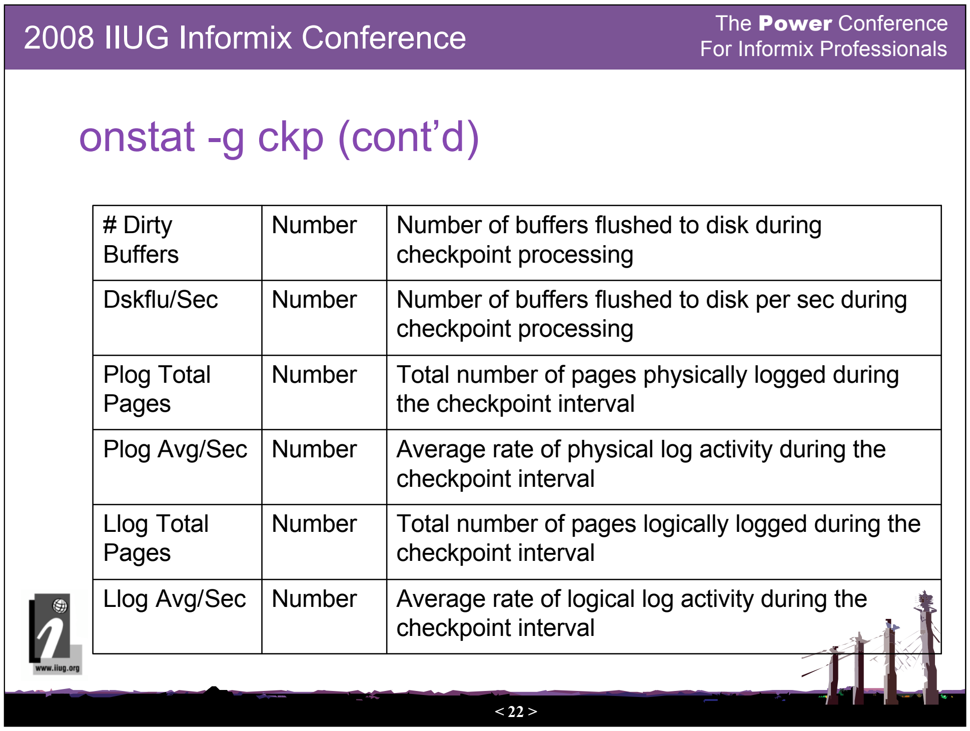
The **Power** Conference
For Informix Professionals

onstat -g ckp (cont'd)

Total Time	Seconds	**Total checkpoint duration from request time to checkpoint completion**
Flush Time	Seconds	**Time to flush bufferpools**
Block Time	Seconds	**Transaction blocking time**
# Waits	Number	Number of transactions that blocked waiting for checkpoint
Ckpt Time	Seconds	Amount of time it takes for all transactions to recognize a checkpoint has been requested
Wait Time	Seconds	Average time thread waited for checkpoint
Long Time	Seconds	Longest amount of time a transaction waited for checkpoint

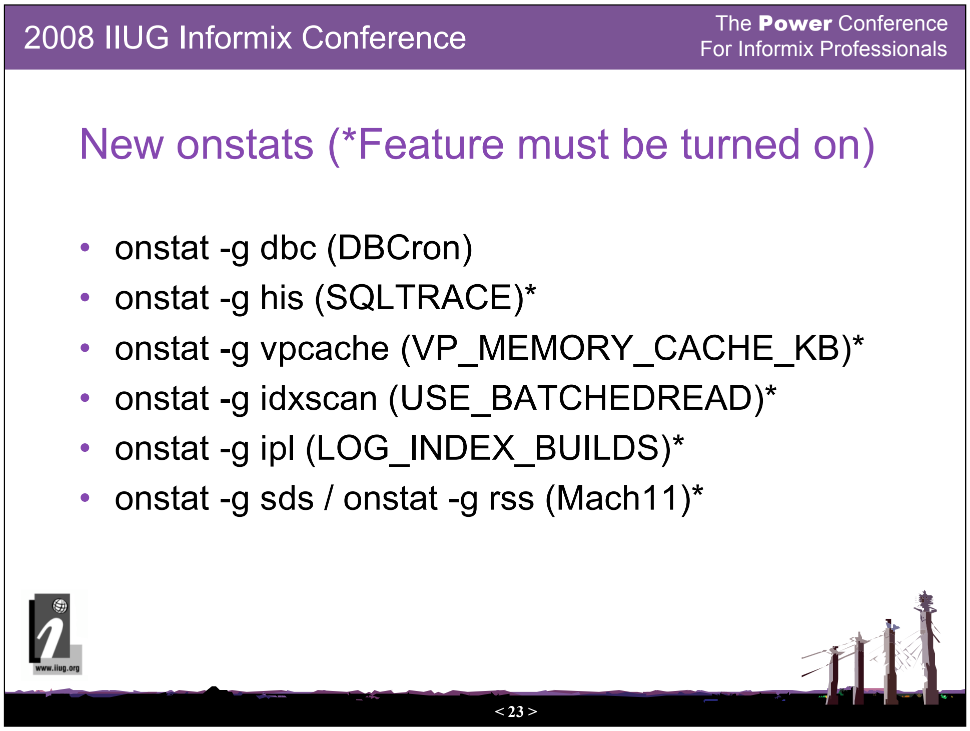
< 21 >

Note that Ckpt Time is simply the amount of time it takes for all transactions to **recognize** that a checkpoint has been requested. The following information may be more useful in determining the total effect of a checkpoint -- “Total Time”, “Flush Time”, and “Block Time”.



onstat -g ckp (cont'd)

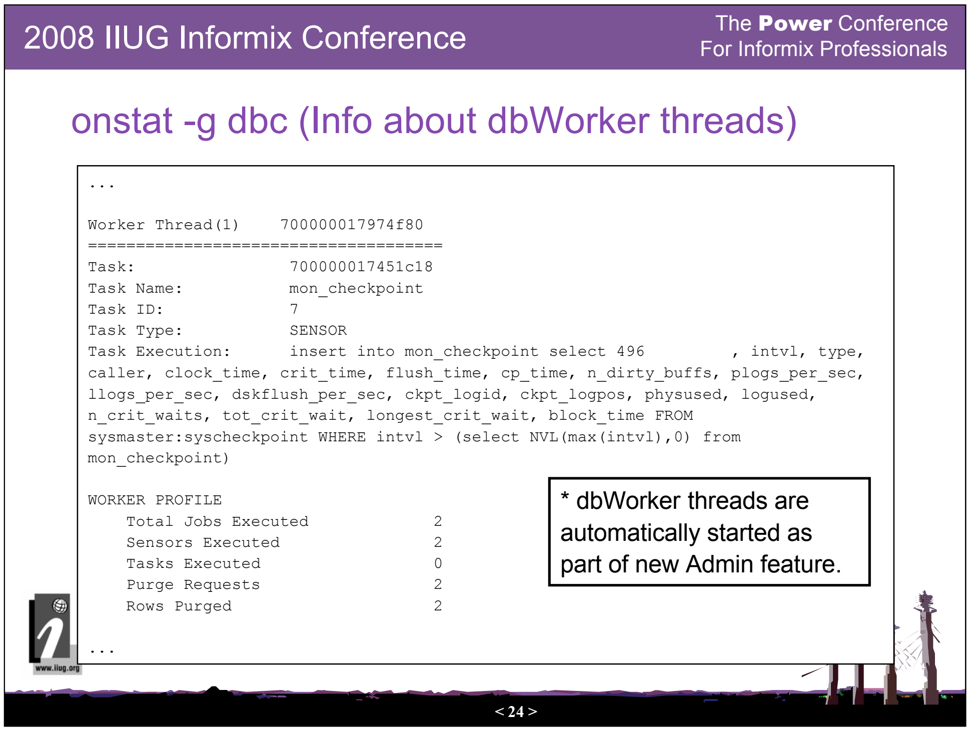
# Dirty Buffers	Number	Number of buffers flushed to disk during checkpoint processing
Dskflu/Sec	Number	Number of buffers flushed to disk per sec during checkpoint processing
Plog Total Pages	Number	Total number of pages physically logged during the checkpoint interval
Plog Avg/Sec	Number	Average rate of physical log activity during the checkpoint interval
Llog Total Pages	Number	Total number of pages logically logged during the checkpoint interval
Llog Avg/Sec	Number	Average rate of logical log activity during the checkpoint interval



New onstats (*Feature must be turned on)

- onstat -g dbc (DBCron)
- onstat -g his (SQLTRACE)*
- onstat -g vpcache (VP_MEMORY_CACHE_KB)*
- onstat -g idxscan (USE_BATCHEDREAD)*
- onstat -g ipl (LOG_INDEX_BUILDS)*
- onstat -g sds / onstat -g rss (Mach11)*

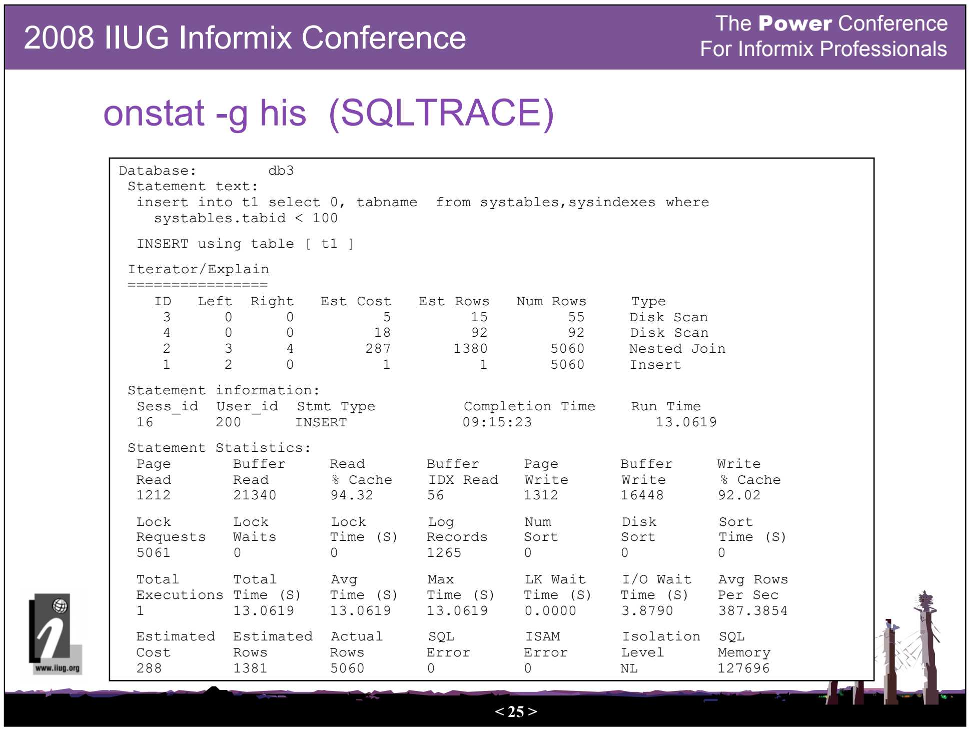
www.iiug.org



onstat -g dbc (Info about dbWorker threads)

```
...

Worker Thread(1)    700000017974f80
=====================================
Task:                    700000017451c18
Task Name:               mon_checkpoint
Task ID:                 7
Task Type:               SENSOR
Task Execution:          insert into mon_checkpoint select 496         , intvl, type,
caller, clock_time, crit_time, flush_time, cp_time, n_dirty_buffs, plogs_per_sec,
llogs_per_sec, dskflush_per_sec, ckpt_logid, ckpt_logpos, physused, logused,
n_crit_waits, tot_crit_wait, longest_crit_wait, block_time FROM
sysmaster:syscheckpoint WHERE intvl > (select NVL(max(intvl),0) from
mon_checkpoint)

WORKER PROFILE
    Total Jobs Executed             2
    Sensors Executed                2
    Tasks Executed                  0
    Purge Requests                  2
    Rows Purged                     2

...
```

* dbWorker threads are automatically started as part of new Admin feature.



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

onstat -g his (SQLTRACE)

```
Database:        db3
 Statement text:
  insert into t1 select 0, tabname  from systables,sysindexes where
    systables.tabid < 100

  INSERT using table [ t1 ]

 Iterator/Explain
 ================
    ID   Left  Right   Est Cost   Est Rows   Num Rows      Type
     3      0      0          5         15         55     Disk Scan
     4      0      0         18         92         92     Disk Scan
     2      3      4        287       1380       5060     Nested Join
     1      2      0          1          1       5060     Insert

 Statement information:
  Sess_id  User_id  Stmt Type          Completion Time     Run Time
  16       200      INSERT             09:15:23               13.0619

 Statement Statistics:
  Page       Buffer     Read       Buffer     Page       Buffer     Write
  Read       Read       % Cache    IDX Read   Write      Write      % Cache
  1212       21340      94.32      56         1312       16448      92.02

  Lock       Lock       Lock       Log        Num        Disk       Sort
  Requests   Waits      Time (S)   Records    Sort       Sort       Time (S)
  5061       0          0          1265       0          0          0

  Total      Total      Avg        Max        LK Wait    I/O Wait   Avg Rows
  Executions Time (S)   Time (S)   Time (S)   Time (S)   Time (S)   Per Sec
  1          13.0619    13.0619    13.0619    0.0000     3.8790     387.3854

  Estimated  Estimated  Actual     SQL        ISAM       Isolation  SQL
  Cost       Rows       Rows       Error      Error      Level      Memory
  288        1381       5060       0          0          NL         127696
```

The SQL Tracing feature, available from IDS 11.10, provides a facility to capture SQL statement history information. The information captured from SQL Tracing includes data on how much time was spent on each leg of the SQL statement execution process, what resources were used to execute the statement, and how long the entire process was. When turned on, this feature uses a fixed amount of memory as a circular buffer to store the information gathered. By default, each statement is allocated a fixed amount of memory to store statistical information about the statement execution. This individual trace item is called the trace buffer. Since the trace buffer is a fixed size, if the SQL statement is large, some of its information can be truncated. The statement information that may be truncated will be the statement text, table names, database name, or the execution iterator information.

The SQL Tracing feature can be enabled and disabled at any time. The number and size of the trace buffers may be resized while the system is running. However, resizing the trace buffer will cause the previous contents of the buffer to be deleted. Since each trace buffer contains the information for a single SQL statement, the number of trace buffers determines how many SQL statements can be traced.

There are two different tracing modes (“user” and “global”) and three levels of tracing (“low”, “med”, “high”). The tracing mode can be enabled for the entire database server or at a user level. The three escalating levels include the prior level’s information and new information. Turning off the SQL Tracing will disable tracing and release all memory associated with the tracing.

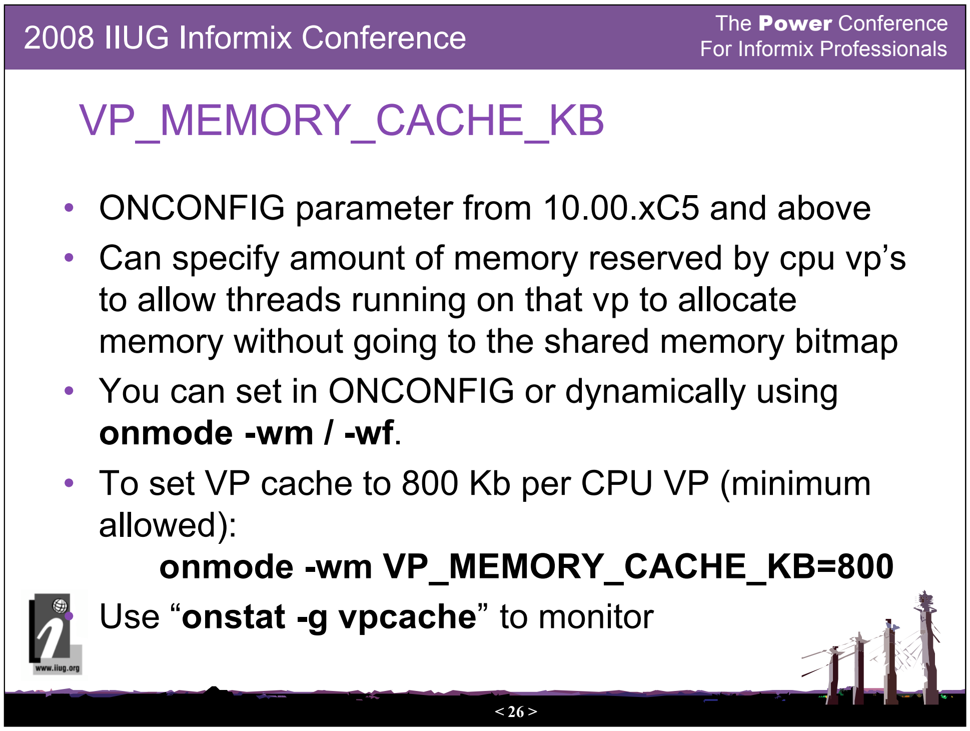
The SQL Tracing feature can be enabled by using the SQLTRACE ONCONFIG parameter or dynamically, by using the sysadmin commands. Here are two examples of enabling the SQL Tracing feature:

1) Using the ONCONFIG SQLTRACE parameter (note that size is specified in kilobytes):

SQLTRACE level=MED,ntraces=2000,size=2,mode=global

2) Using the sysadmin task function:

EXECUTE FUNCTION task(“set sql tracing on”, 2000, “2k”, “med”, “global”);



2008 IIUG Informix Conference

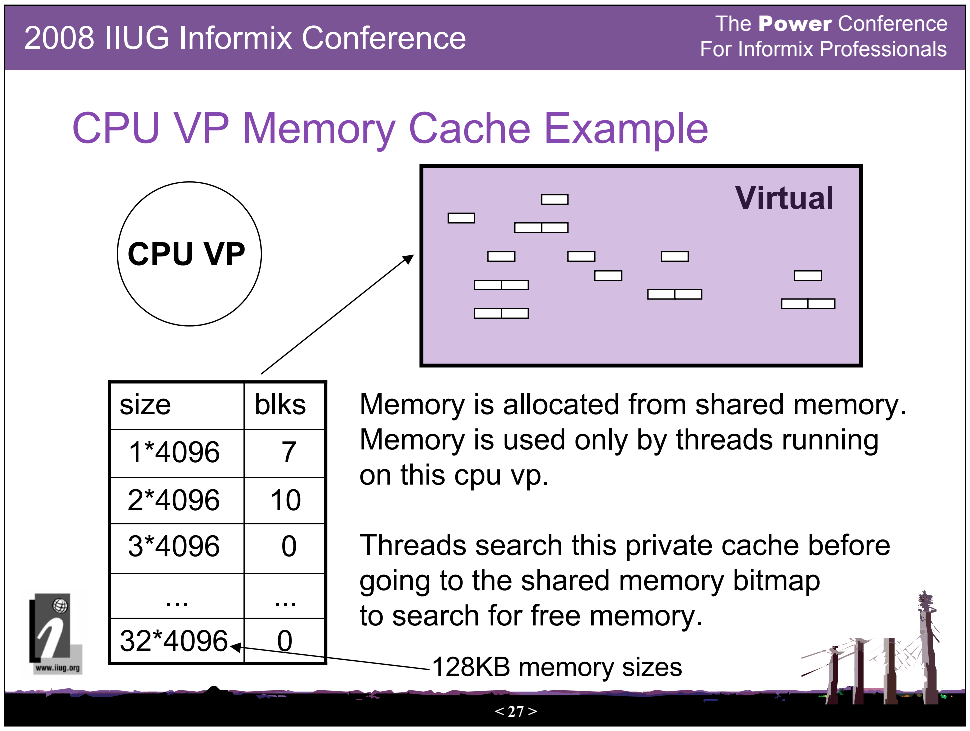
The **Power** Conference
For Informix Professionals

VP_MEMORY_CACHE_KB

- ONCONFIG parameter from 10.00.xC5 and above
- Can specify amount of memory reserved by cpu vp's to allow threads running on that vp to allocate memory without going to the shared memory bitmap
- You can set in ONCONFIG or dynamically using **onmode -wm / -wf**.
- To set VP cache to 800 Kb per CPU VP (minimum allowed):

  onmode -wm VP_MEMORY_CACHE_KB=800
- Use "**onstat -g vpcache**" to monitor

ONCONFIG parameter VP_MEMORY_CACHE_KB -- specifies the amount of private memory blocks of your CPU VP, in KB, that the database server can access. The cpu vp private memory is allocated from IDS shared memory. Acceptable values are: 0 (disable), 800 through 40% of the value of SHMTOTAL.



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

CPU VP Memory Cache Example

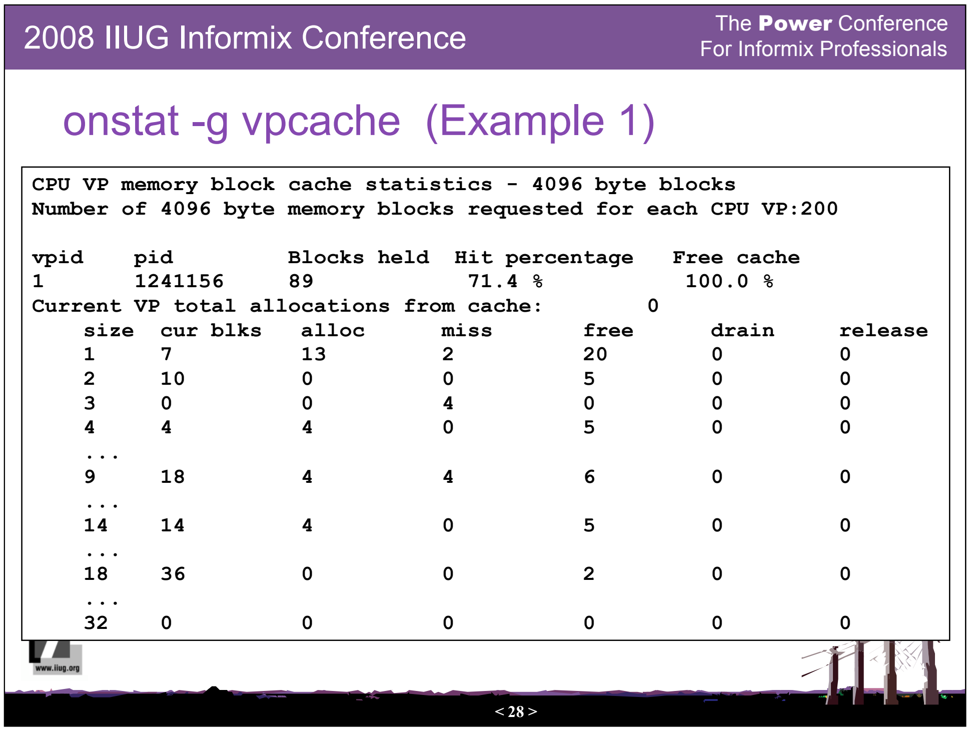
size	blks
1*4096	7
2*4096	10
3*4096	0
...	...
32*4096	0

Memory is allocated from shared memory.
Memory is used only by threads running on this cpu vp.

Threads search this private cache before going to the shared memory bitmap to search for free memory.

128KB memory sizes

```
VP_MEMORY_CACHE_KB=800
```



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

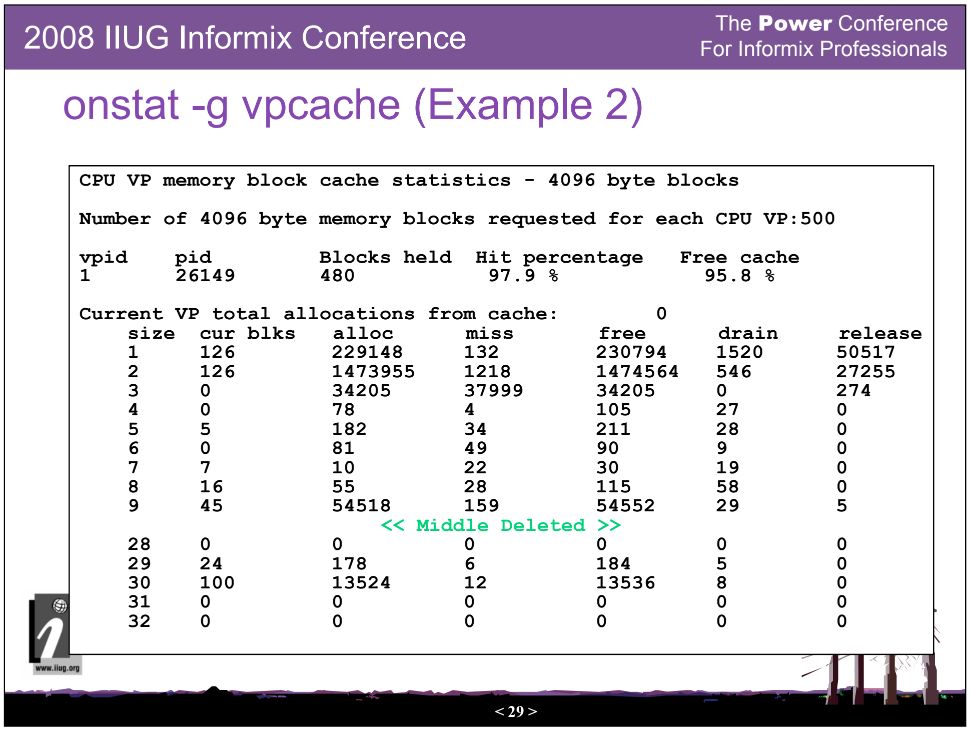
onstat -g vpcache (Example 1)

```
CPU VP memory block cache statistics - 4096 byte blocks
Number of 4096 byte memory blocks requested for each CPU VP:200

vpid    pid         Blocks held  Hit percentage   Free cache
1       1241156     89             71.4 %           100.0 %
Current VP total allocations from cache:         0
    size  cur blks   alloc      miss       free      drain     release
    1     7          13         2          20        0         0
    2     10         0          0          5         0         0
    3     0          0          4          0         0         0
    4     4          4          0          5         0         0
    ...
    9     18         4          4          6         0         0
    ...
    14    14         4          0          5         0         0
    ...
    18    36         0          0          2         0         0
    ...
    32    0          0          0          0         0         0
```

< 28 >

VP_MEMORY_CACHE_KB=800

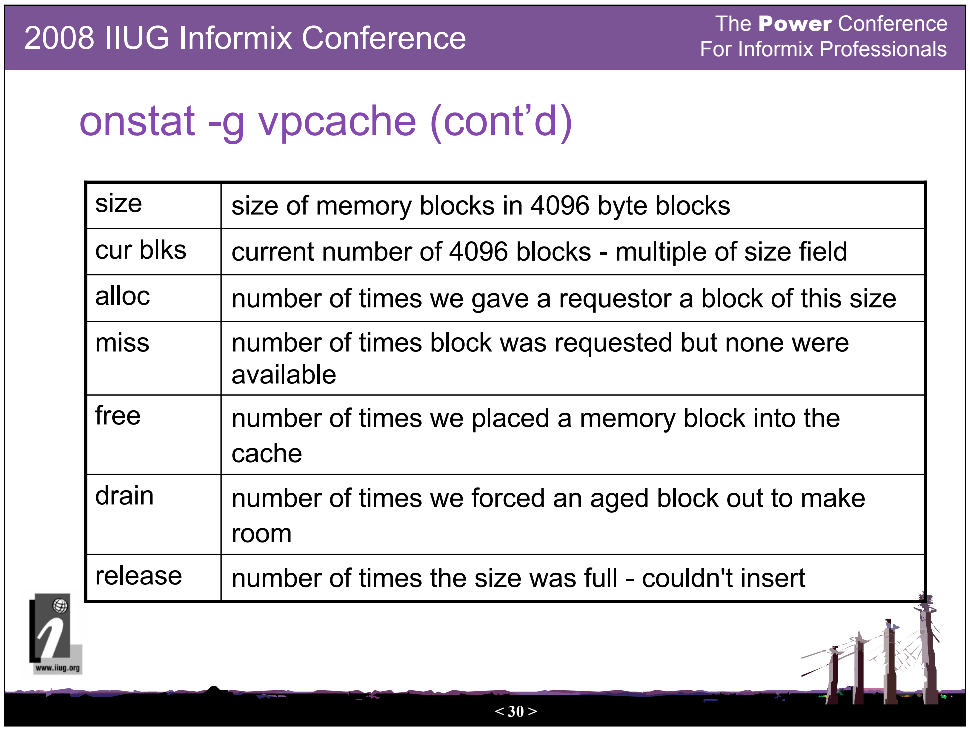
Each cpu vp has a private memory cache of 800 KB, that translates to 800KB/4KB = 200 blocks of 4096 byte memory blocks that can be allocated for each cpu vp.



onstat -g vpcache (Example 2)

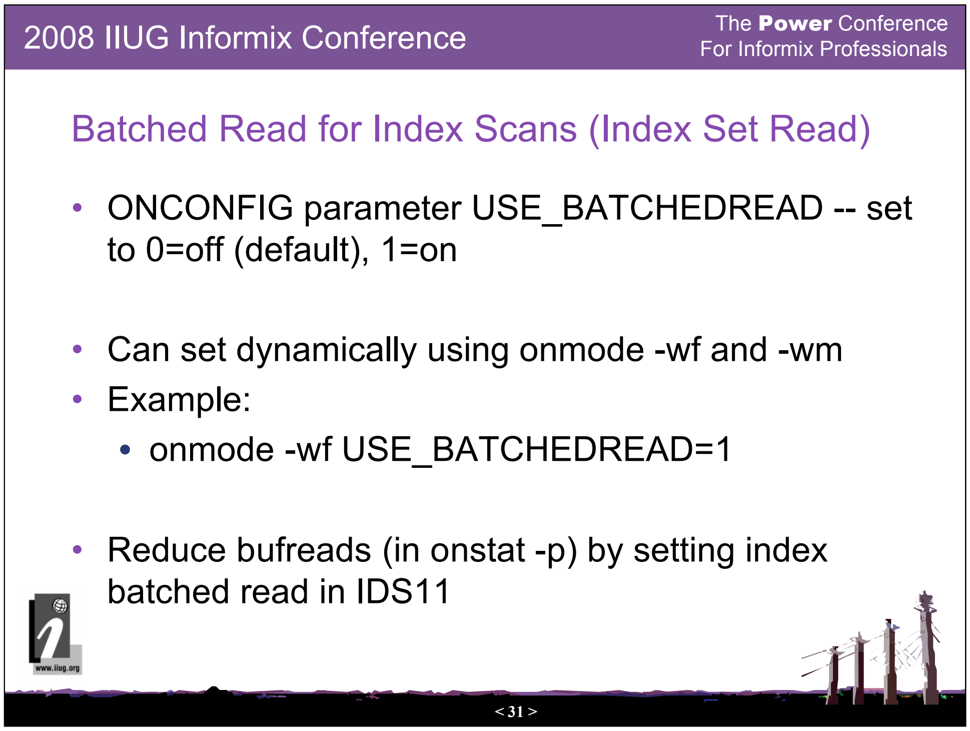
```
CPU VP memory block cache statistics - 4096 byte blocks

Number of 4096 byte memory blocks requested for each CPU VP:500

vpid    pid         Blocks held  Hit percentage   Free cache
1       26149       480           97.9 %            95.8 %

Current VP total allocations from cache:        0
    size  cur blks   alloc      miss       free      drain     release
    1     126        229148     132        230794    1520      50517
    2     126        1473955    1218       1474564   546       27255
    3     0          34205      37999      34205     0         274
    4     0          78         4          105       27        0
    5     5          182        34         211       28        0
    6     0          81         49         90        9         0
    7     7          10         22         30        19        0
    8     16         55         28         115       58        0
    9     45         54518      159        54552     29        5
                         << Middle Deleted >>
    28    0          0          0          0         0         0
    29    24         178        6          184       5         0
    30    100        13524      12         13536     8         0
    31    0          0          0          0         0         0
    32    0          0          0          0         0         0
```

VP_MEMORY_CACHE_KB=2000



onstat -g vpcache (cont’d)

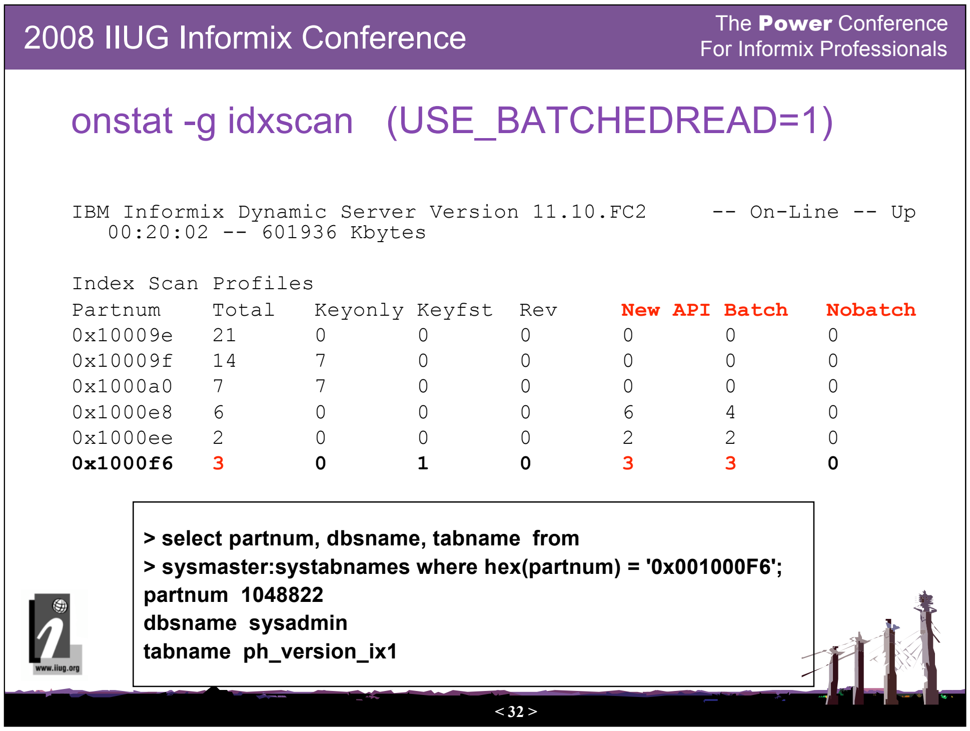
size	size of memory blocks in 4096 byte blocks
cur blks	current number of 4096 blocks - multiple of size field
alloc	number of times we gave a requestor a block of this size
miss	number of times block was requested but none were available
free	number of times we placed a memory block into the cache
drain	number of times we forced an aged block out to make room
release	number of times the size was full - couldn't insert



Batched Read for Index Scans (Index Set Read)

- ONCONFIG parameter USE_BATCHEDREAD -- set to 0=off (default), 1=on
- Can set dynamically using onmode -wf and -wm
- Example:
  - onmode -wf USE_BATCHEDREAD=1
- Reduce bufreads (in onstat -p) by setting index batched read in IDS11

USE_BATCHEDREAD for index scans was introduced in IDS 11 (11.10). Can monitor with onstat -g idxscan and onstat -p (look at bufreads).



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

onstat -g idxscan   (USE_BATCHEDREAD=1)

```
IBM Informix Dynamic Server Version 11.10.FC2     -- On-Line -- Up
   00:20:02 -- 601936 Kbytes

Index Scan Profiles
Partnum     Total    Keyonly Keyfst  Rev      New API  Batch    Nobatch
0x10009e    21       0       0       0        0        0        0
0x10009f    14       7       0       0        0        0        0
0x1000a0    7        7       0       0        0        0        0
0x1000e8    6        0       0       0        6        4        0
0x1000ee    2        0       0       0        2        2        0
0x1000f6    3        0       1       0        3        3        0
```

```
> select partnum, dbsname, tabname  from
> sysmaster:systabnames where hex(partnum) = '0x001000F6';
partnum  1048822
dbsname  sysadmin
tabname  ph_version_ix1
```

Even when USE_BATCHEDREAD for index scans is set, it does not mean that the new index scan API will be used. So, the column “New API” shows whether the new index scan method is being used, and if so, how much of the index reads are done in “Batch”, and how many are not.

Enabling USE_BATCHEDREAD should significantly reduce the number of bufreads (as shown in onstat -p) during index scans.

Example:

create table t(c int);

insert into t values (1);

insert into t values (2);

create index i on t(c)

onmde -wf USE_BATCHEDREAD=0

onstat -z

select c {+index i,c} from t where c < 3;

onstat -p

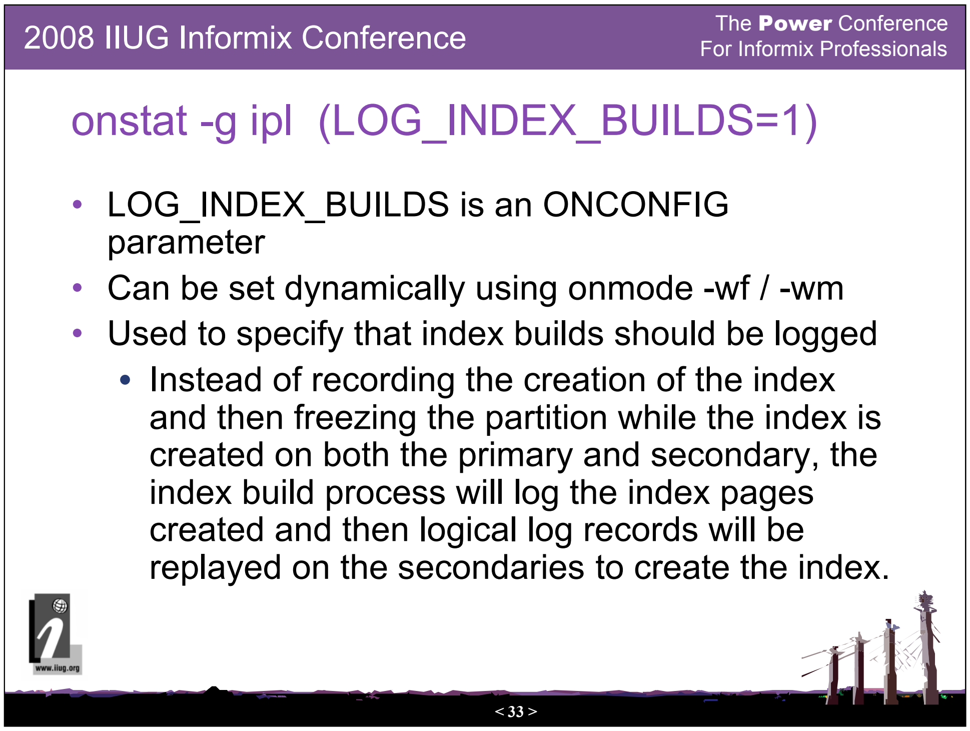
onmde -wf USE_BATCHEDREAD=1

onstat -z

select c {+index i,c} from t where c < 3;

onstat -p

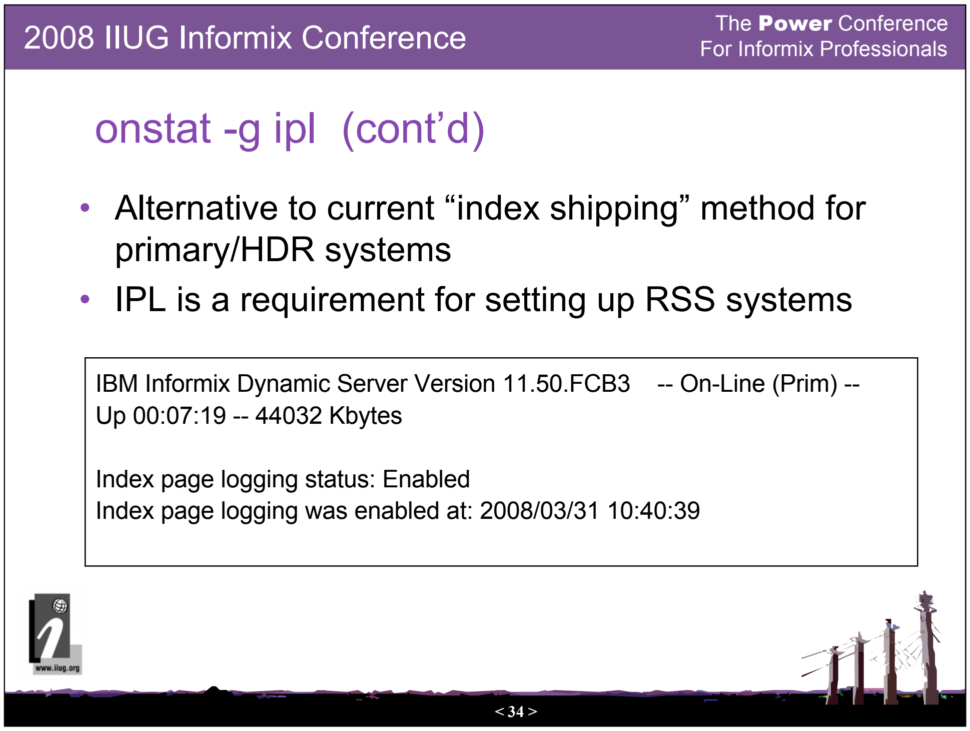
The number of bufreads should be lower with USE_BATCHEDREAD set to 1.



onstat -g ipl (LOG_INDEX_BUILDS=1)

- LOG_INDEX_BUILDS is an ONCONFIG parameter
- Can be set dynamically using onmode -wf / -wm
- Used to specify that index builds should be logged
  - Instead of recording the creation of the index and then freezing the partition while the index is created on both the primary and secondary, the index build process will log the index pages created and then logical log records will be replayed on the secondaries to create the index.

When setting up HDR, you can choose between “index page logging” or “index shipping”.



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

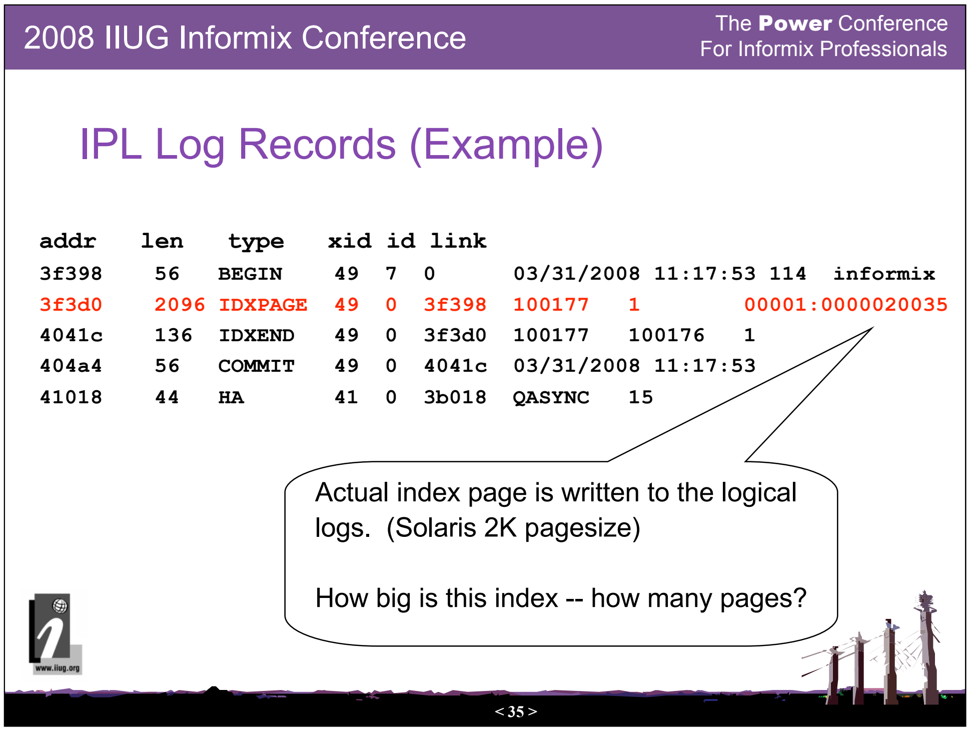
onstat -g ipl (cont'd)

- Alternative to current "index shipping" method for primary/HDR systems
- IPL is a requirement for setting up RSS systems

```
IBM Informix Dynamic Server Version 11.50.FCB3    -- On-Line (Prim) --
Up 00:07:19 -- 44032 Kbytes

Index page logging status: Enabled
Index page logging was enabled at: 2008/03/31 10:40:39
```

You can also select from sysmaster:sysipl -- both ipl_status and ipl_time are defined as integers.

```
> select * from sysmaster:sysipl;


 ipl_status    ipl_time


          1  1206985239


1 row(s) retrieved.
```



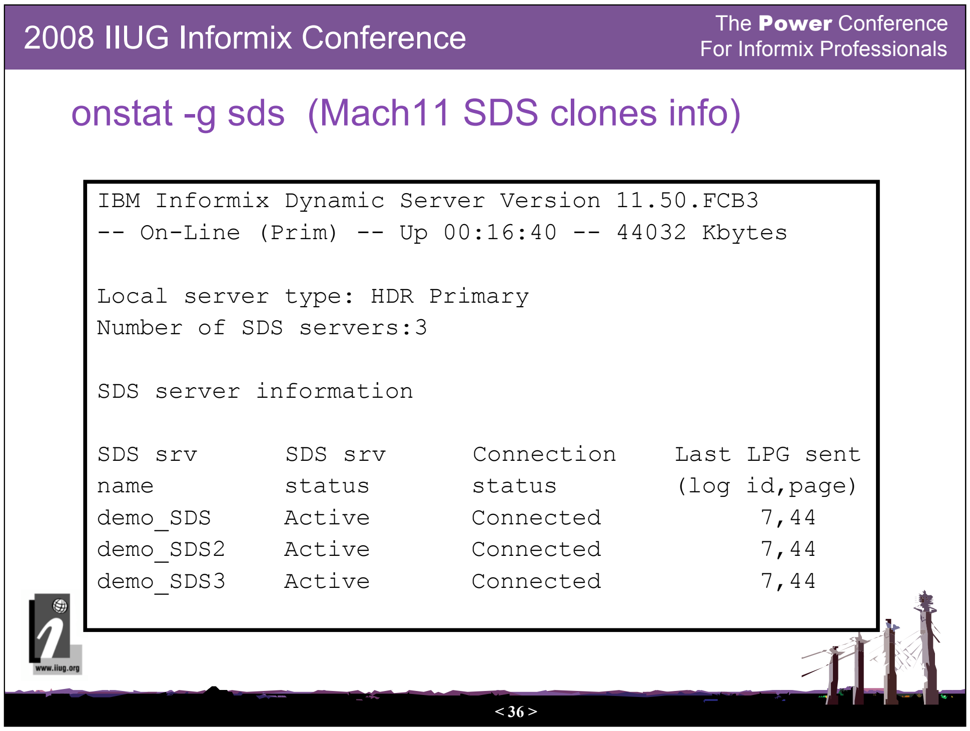
IPL Log Records (Example)

```
addr    len   type     xid id link
3f398    56   BEGIN     49  7  0      03/31/2008 11:17:53 114  informix
3f3d0    2096 IDXPAGE   49  0  3f398  100177   1         00001:0000020035
4041c    136  IDXEND    49  0  3f3d0  100177   100176   1
404a4    56   COMMIT    49  0  4041c  03/31/2008 11:17:53
41018    44   HA        41  0  3b018  QASYNC   15
```

Actual index page is written to the logical logs. (Solaris 2K pagesize)

How big is this index -- how many pages?

onlog -n <log #>

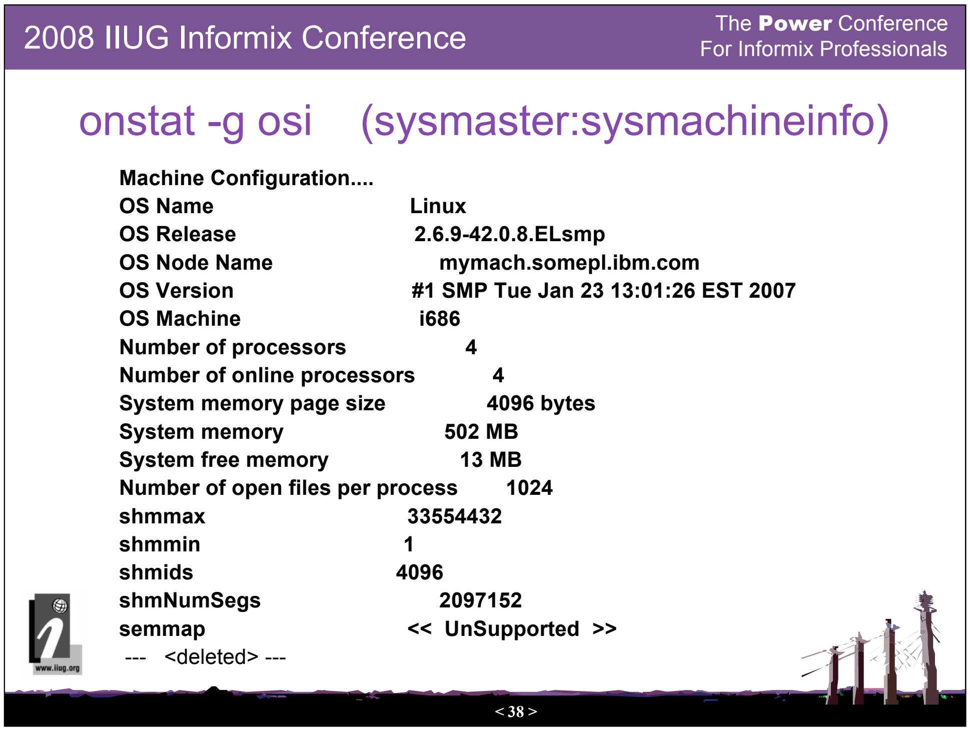


onstat -g sds (Mach11 SDS clones info)

```
IBM Informix Dynamic Server Version 11.50.FCB3
-- On-Line (Prim) -- Up 00:16:40 -- 44032 Kbytes

Local server type: HDR Primary
Number of SDS servers:3

SDS server information

SDS srv      SDS srv      Connection     Last LPG sent
name         status       status         (log id,page)
demo_SDS     Active       Connected           7,44
demo_SDS2    Active       Connected           7,44
demo_SDS3    Active       Connected           7,44
```

onstat -g rss (Mach11 RSS clones info)

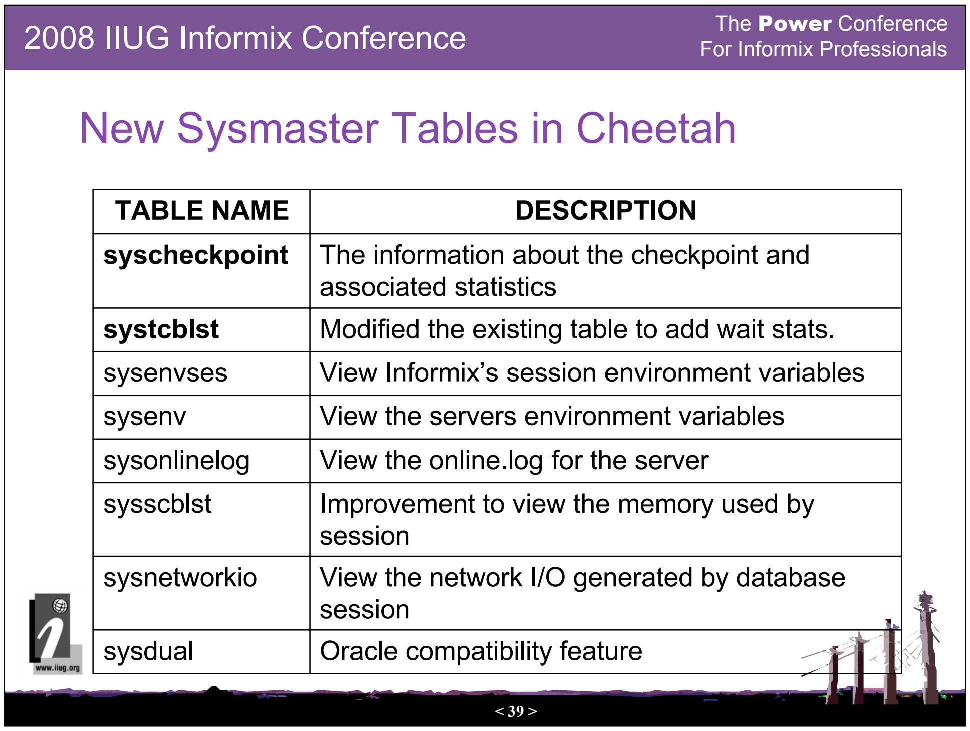
```
Local server type: HDR Primary
Index page logging status: Enabled
Index page logging was enabled at: 2008/03/31 10:40:39
Number of RSS servers: 4

RSS Server information:
RSS Srv      RSS Srv      Connection     Next LPG to send
name         status       status           (log id,page)
demo_RSS     Active       Connected               7,46
demo_RSS2    Active       Connected               7,46
demo_RSS3    Active       Connected               7,46
demo_RSS4    Active       Connected               7,46
```



onstat -g osi   (sysmaster:sysmachineinfo)

```
Machine Configuration....
OS Name                            Linux
OS Release                         2.6.9-42.0.8.ELsmp
OS Node Name                          mymach.somepl.ibm.com
OS Version                         #1 SMP Tue Jan 23 13:01:26 EST 2007
OS Machine                          i686
Number of processors                     4
Number of online processors              4
System memory page size                4096 bytes
System memory                       502 MB
System free memory                    13 MB
Number of open files per process       1024
shmmax                            33554432
shmmin                            1
shmids                           4096
shmNumSegs                            2097152
semmap                            <<  UnSupported  >>
 ---   <deleted> ---
```

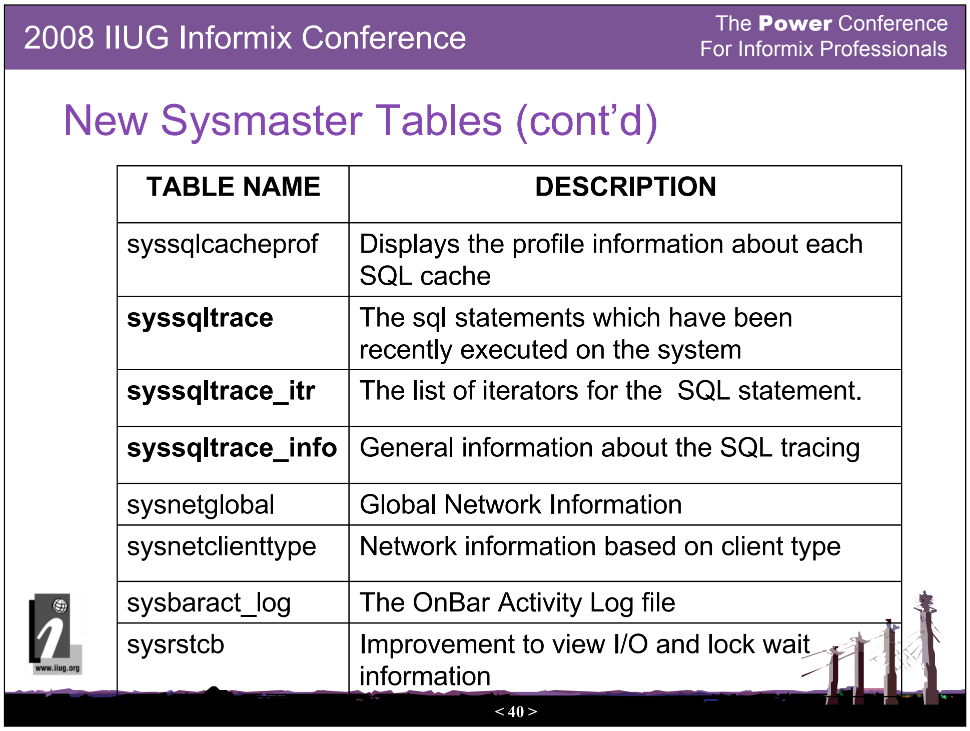
onstat -g osi is a useful onstat that displays common machine configuration info and kernel parameters. The same information can also be obtained from sysmaster:sysmachineinfo pseudo table.



New Sysmaster Tables in Cheetah

TABLE NAME	DESCRIPTION
syscheckpoint	The information about the checkpoint and associated statistics
systcblst	Modified the existing table to add wait stats.
sysenvses	View Informix's session environment variables
sysenv	View the servers environment variables
sysonlinelog	View the online.log for the server
sysscblst	Improvement to view the memory used by session
sysnetworkio	View the network I/O generated by database session
sysdual	Oracle compatibility feature

There are many new (or enhancements to existing sysmaster pseudo tables) -- these are some examples.



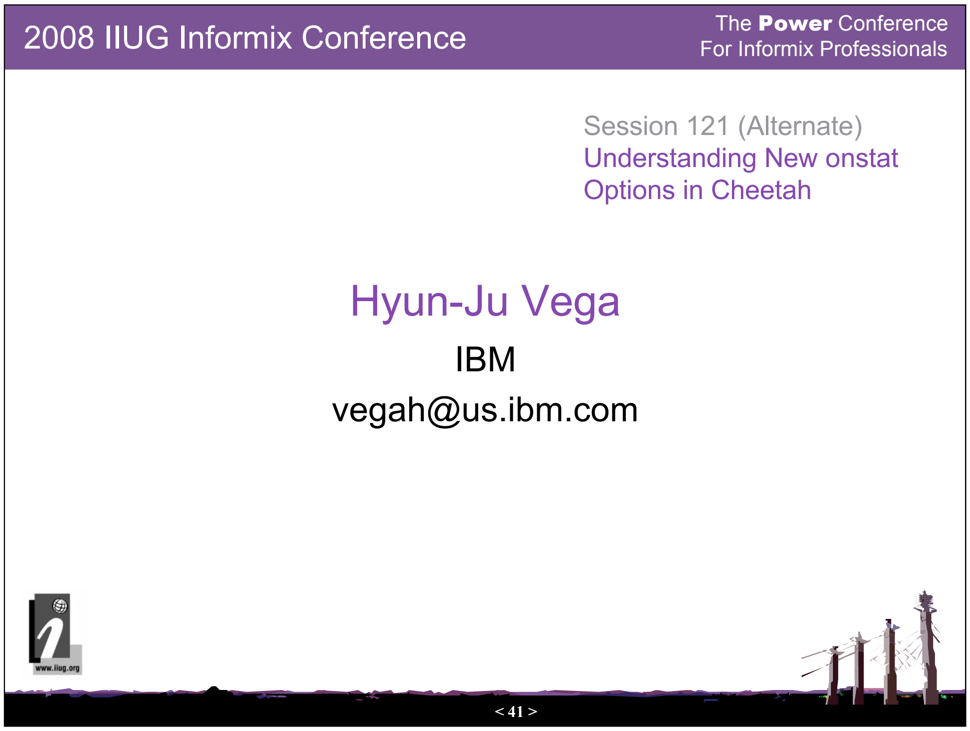
2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

New Sysmaster Tables (cont'd)

TABLE NAME	DESCRIPTION
syssqlcacheprof	Displays the profile information about each SQL cache
syssqltrace	The sql statements which have been recently executed on the system
syssqltrace_itr	The list of iterators for the SQL statement.
syssqltrace_info	General information about the SQL tracing
sysnetglobal	Global Network Information
sysnetclienttype	Network information based on client type
sysbaract_log	The OnBar Activity Log file
sysrstcb	Improvement to view I/O and lock wait information

There are many new (or enhancements to existing sysmaster pseudo tables) -- these are some examples.



Session 121 (Alternate)

Understanding New onstat Options in Cheetah

Hyun-Ju Vega

IBM

vegah@us.ibm.com