IDS Autonomic Tuning

Scott Pickett
WW Informix Technical Sales
For questions about this presentation contact: spickett@us.ibm.com

Overview

- **Problem Statement:**
  - We want the Informix database server to be as invisible as possible.
  - After the initial install of IDS and uptime, we need the server to be able to adapt to any workload that comes its way.
  - Prior to 12.10xC2, several key server resources were not dynamic
    - This could and did lead to artificial performance ceilings, and an increase in downtime to fix, or degradation in performance until the issue was detected by human interface.
      - Not everyone knows how to detect ……. and if they do, getting the downtime from management is an issue …. always.

- **The improvements are in several different categories…..**

Requirement Categories

- **Physical Log**
- **Logical Log**
- **Bufferpool**
- **CPUs**

- **These will follow similar rules guiding automatic tuning:**
  - How to detect when insufficient resources for each of the above resources are causing a performance impact
  - How to increase the resource with minimal impact to the application
  - How to control the increase of the resource so it adapts in a timely manner… not too much, not too little, just right

Physical Log Expansion

- **Purpose:**
  - To help avoid performance loss associated with an undersized physical log caused by frequent server checkpoints which can block user transactions, located within an already constrained storage space with other objects therein, such as database and server objects.

- **Expansion**
  - We support the changing of the physical log location already.
  - We want to make sure there is only sole occupancy in the space for the physical log; nothing else hindering its successful expansion and requested size:
    - A new single Physical Log Storage Space (PLOGspace) per instance,
    - Has a single physical chunk containing only the server physical log with the physical page size of the host O/S.

- **Physical log expansion is triggered by any of the following**
  - A checkpoint blocked user transactions because the physical log is too small.
  - 5 checkpoints triggered by physical log activity:
    - Since the last time the physical log was extended.
    - And within the last 20 checkpoints.

Physical Log Space Creation – onspaces

- **General Syntax:**

  onspaces -c –P <name> -p <path> -o <offset> -s <size> [-m <mpath> <offset>]

- **Examples:**

  onspaces -c –P plogdbs -p /dev/chk1 -o 0 -s 40000
  onspaces -c –P pdbs1 -p /dev/pchk1 -o 500 -s 60000 -m /dev/mchk1 0

- **Note**
  - If a PLOGspace already exists:
    - Creating a new one will create the new PLOGspace,
    - Move the physical log to that space,
    - Drop the old PLOGspace,
    - All of the above done automatically.

Physical Log Space Creation – SQL Admin API

- **Creating a PLOGspace with SQL Admin API**

- **General Syntax**:

  create plogspace, <name>, <path>, <size>, <offset>, <mir path>, <mir offset>

- **Examples:**

  execute function task("create plogspace","plogdbs","/dev/chk1",30000,0);
  execute function task("create plogspace","plogdbs","/dev/chk1",30000,0,"/dev/mchk1",0);

Physical Log Improvements - onspaces

- **Dropping a PLOGspace with onspaces**
  - The same space-dropping command (**onspaces –d**) may be used to drop this new type of space,
  - Only if the space is empty (53 pages, **oncheck –pe** to verify)

- **General Syntax:**

  onspaces -d <PLOGspace name> [-y]

- **Example:**

  onspaces -d plogdbs -y

Physical Log Improvements – SQL Admin API

- **Dropping a PLOGspace with SQL Admin**
  - One must use a new "**drop plogspace**" command with SQL Admin API.
    - "**drop dbspace**" will return an error.
  - "**drop plogspace**" takes no arguments, since there can be at most one PLOGspace in the instance.
    - The PLOGspace must be empty to be dropped.

- **General Syntax:**

  drop plogspace

- **Example:**

  execute function task("drop plogspace");

Warnings on Insufficient PLOGspace

- **In the message log file:**
  - Thru the **onstat –m** command line utility:
    - You will see a message saying “Physical Log too small ……”
- **Thru onstat –g ckp utility:**
  - Look for a warning at the top about the physical log being too small or a “Trigger” of “Plog” with “Block Time” > 0.0
  - The presence of long transaction rollbacks. A “Trigger” of “LongTX” with “Block Time” > 0.0 can be a too small physical log or too little Logical Log space.

```
informix@cte2:~> onstat -g ckp

IBM Informix Dynamic Server Version 12.10.FC3 -- On-Line -- Up 2 days 00:56:16 -- 240192 Kbytes

AUTO_CKPTS=Off   RTO_SERVER_RESTART=60 seconds   Estimated recovery time 15 seconds
```

							Critical Sections					Physical Log		Logical Log	
Interval	Clock 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
42	15:19:41	*Admin	4:0x569108	0.0	0.0	0.0	1	0.0	0.0	0.0	0	0	2	2	2	2
43	15:19:41	*Admin	4:0x56b108	0.0	0.0	0.0	1	0.0	0.0	0.0	0	0	2	2	2	2
44	15:19:42	*Admin	4:0x56e288	0.0	0.0	0.0	1	0.0	0.0	0.0	1	1	7	7	3	3
45	15:19:42	*Admin	4:0x571288	0.1	0.0	0.0	1	0.0	0.1	0.1	1	1	10	10	3	3
46	15:19:42	*Admin	4:0x574288	0.0	0.0	0.0	1	0.0	0.0	0.0	1	1	9	9	3	3
47	15:19:45	*User	5:0x2018	0.0	0.0	0.0	1	0.0	0.0	0.0	1	1	8	2	6	2
48	15:19:46	*Admin	5:0x5288	0.0	0.0	0.0	1	0.0	0.0	0.0	1	1	9	9	3	3
49	15:19:47	*User	5:0xa018	0.0	0.0	0.0	1	0.0	0.0	0.0	1	1	8	8	5	5
50	15:19:53	Admin	5:0x74018	0.2	0.2	0.0	1	0.0	0.0	0.0	77	77	65	10	106	17
51	15:39:12	LongTX	13:0x6c	3.9	3.7	3.8	1	0.0	3.8	3.8	6516	1764	270	0	26044	22
52	15:39:13	Admin	13:0x20b4	1.3	1.3	0.0	3	0.0	0.6	1.1	1	0	42	10	165	41
53	15:39:13	*Admin	13:0x1750b8	0.2	0.0	0.0	2	0.0	0.2	0.3	29	29	19	19	371	371
54	15:53:58	LongTX	17:0x84	4.8	4.7	4.8	1	0.0	4.8	4.8	5896	1252	2085	2	13021	14
55	15:54:00	Admin	17:0x20b4	1.5	1.5	0.0	3	0.0	0.8	1.2	1	0	21	4	166	33
56	15:54:01	*Admin	17:0x1d4420	1.0	1.0	0.0	2	0.0	1.0	1.0	227	227	172	86	466	233
57	14:50:08	*Admin	20:0x2e33b4	0.8	0.0	0.0	1	0.0	0.7	0.7	65	65	354	0	10081	0
58	14:50:11	*Admin	20:0x2e75a8	2.0	0.0	0.0	1	0.0	2.0	2.0	4	4	20	6	4	1
59	14:50:14	*Admin	20:0x2ed2e4	0.0	0.0	0.0	1	0.0	0.0	0.0	1	1	6	2	6	2
60	14:50:14	Admin	20:0x2f0080	0.0	0.0	0.0	1	0.0	0.0	0.0	0	0	4	4	4	4
61	14:50:32	*CDR	21:0x49080	0.1	0.0	0.0	1	0.0	0.1	0.1	91	91	112	6	2591	143

Logical Log Improvements

- **Purpose**
  - To improve performance, having sufficient logical log space helps prevent frequent checkpoints from blocking transactions and also from having long or too frequent checkpoints.
  - Knowing an appropriate amount of logical log space for a dynamic workload is difficult to supply in advance without first knowing and estimating the transaction load.
  - Configuring a server with too much logical log space in advance can limit the out-of-box experience by requiring substantial disk space.
  - We will therefore allocate logical log space dynamically as needed and therefore dynamically tunable.

- **Detection**
  - The best way to detect if the logical log is a bottleneck is to look at the recent checkpoint activity:
    - The server keeps track of the last 20 checkpoints.
    - If a substantial portion of the recent checkpoints were triggered because of the logical log, then we should increase the logical log space by adding another logical log.
    - If a long transaction or checkpoint blocking occurs because of the logical log, we should also add another logical log to the log space.

Logical Log Improvements

- **Expansion**
  - We already have the adding logical logs dynamically feature:
    - **DYNAMIC_LOGS 2** in the configuration file defined by **$ONCONFIG** environment variable.
      - Typically executes only when the server runs out of logical log space.
    - The server already contains the logic to extend a dbspace while adding a logical log.
  - There is a new, additional Logical Log expansion parameter in the configuration file called **AUTO_LLOG**
    - Enables the automatic addition of logical logs when the database server detects that adding logical logs improves performance.
    - If enabled, log expansion occurs:
      - When 5 of the last 20 checkpoints were caused by logical logs filling up
      - When a logical log causes a blocking checkpoint
      - When a logical log causes a long checkpoint
    - **AUTO_LLOG** & **DYNAMIC_LOGS** do not interact

Logical Log Improvements

- **Configuration**
  - The new **AUTO_LLOG** configuration parameter specifies
    - Dbspace for new logical log files
    - Maximum size of all logical log files.

- **Estimation of maximum logical log space requirements:**
  - Depending on the number of concurrent users accessing your database server…….
    - 1 - 100: 200 MB
    - 101 - 500: 500 MB
    - 501 - 1000: 1 GB
    - More than 1000: 2 GB

AUTO_LLOG – Syntax

- **onconfig.std value 0**
- **By default this is 0, disabled.**
- **AUTO_LLOG 1, *dbspace_name*, *max_size***
  - **1** - Add Logical logs when needed to improve performance.
  - ***dbspace_name*** - dbspace name in which to add logical log files.
    - This dbspace must have the default page size for the operating system.
  - ***max_size*** = Optional. Default is 2048000 KB (2 GB).
    - The maximum size in KB of all logical log files, including any logical log files that are not stored in the dbspace *dbspace_name*.
    - If **max_size** is not specified, the **AUTO_TUNE_SERVER_SIZE** configuration parameter setting affects the maximum size.
  - Takes effect:
    - After you edit the **onconfig** file and restart the database server.
    - Reset the value dynamically in your **onconfig** file by running the **onmode -wf** command
    - Reset the value in memory only by running the **onmode -wm** command.

AUTO_LLOG – Syntax Notes

- When the maximum size of the logical log files is reached, logical log files are no longer added to improve performance. However, if the DYNAMIC_LOGS configuration parameter is enabled, logical logs are added to prevent transaction blocking.
- The settings of the DYNAMIC_LOGS and the AUTO_LLOG configuration parameters do not interact.
- If the value of the *max_size* field is larger than the size of the specified dbspace, enable automatic expansion of storage spaces.

Warnings on Insufficient Logical Logs

- **In the message log file:**
  - Thru the **onstat –m** command line utility:
    - You may see unusual numbers of logical logs have been added.
- **Thru onstat –g ckp utility:**
  - Look for a “Trigger” of “Llog” with “Block Time” > 0.0

```
informix@cte2:~> onstat -g ckp

IBM Informix Dynamic Server Version 12.10.FC3 -- On-Line -- Up 2 days 00:56:16 -- 240192 Kbytes

AUTO_CKPTS=Off   RTO_SERVER_RESTART=60 seconds   Estimated recovery time 15 seconds
```

Interval	Clock Time	Trigger	LSN	Total Time	Flush Time	Block Time	Critical Sections # Waits	Ckpt Time	Wait Time	Long Time	# Dirty Buffers	Dskflu /Sec	Physical Log Total Pages	Physical Log Avg /Sec	Logical Log Total Pages	Logical Log Avg /Sec
42	15:19:41	*Admin	4:0x569108	0.0	0.0	0.0	1	0.0	0.0	0.0	0	0	2	2	2	2
43	15:19:41	*Admin	4:0x56b108	0.0	0.0	0.0	1	0.0	0.0	0.0	0	0	2	2	2	2
44	15:19:42	*Admin	4:0x56e288	0.0	0.0	0.0	1	0.0	0.0	0.0	1	1	7	7	3	3
45	15:19:42	*Admin	4:0x571288	0.1	0.0	0.0	1	0.0	0.1	0.1	1	1	10	10	3	3
46	15:19:42	*Admin	4:0x574288	0.0	0.0	0.0	1	0.0	0.0	0.0	1	1	9	9	3	3
47	15:19:45	*User	5:0x2018	0.0	0.0	0.0	1	0.0	0.0	0.0	1	1	8	2	6	2
48	15:19:46	*Admin	5:0x5288	0.0	0.0	0.0	1	0.0	0.0	0.0	1	1	9	9	3	3
49	15:19:47	*User	5:0xa018	0.0	0.0	0.0	1	0.0	0.0	0.0	1	1	8	8	5	5
50	15:19:53	Admin	5:0x74018	0.2	0.2	0.0	1	0.0	0.0	0.0	77	77	65	10	106	17
51	15:39:12	LongTX	13:0x6c	3.9	3.7	3.8	1	0.0	3.8	3.8	6516	1764	270	0	26044	22
52	15:39:13	Admin	13:0x20b4	1.3	1.3	0.0	3	0.0	0.6	1.1	1	0	42	10	165	41
53	15:39:13	*Admin	13:0x1750b8	0.2	0.0	0.0	2	0.0	0.2	0.3	29	29	19	19	371	371
54	15:53:58	LongTX	17:0x84	4.8	4.7	4.8	1	0.0	4.8	4.8	5896	1252	2085	2	13021	14
55	15:54:00	Admin	17:0x20b4	1.5	1.5	0.0	3	0.0	0.8	1.2	1	0	21	4	166	33
56	15:54:01	*Admin	17:0x1d4420	1.0	1.0	0.0	2	0.0	1.0	1.0	227	227	172	86	466	233
57	14:50:08	*Admin	20:0x2e33b4	0.8	0.0	0.0	1	0.0	0.7	0.7	65	65	354	0	10081	0
58	14:50:11	*Admin	20:0x2e75a8	2.0	0.0	0.0	1	0.0	2.0	2.0	4	4	20	6	4	1
59	14:50:14	*Admin	20:0x2ed2e4	0.0	0.0	0.0	1	0.0	0.0	0.0	1	1	6	2	6	2
60	14:50:14	Admin	20:0x2f0080	0.0	0.0	0.0	1	0.0	0.0	0.0	0	0	4	4	4	4
61	14:50:32	*CDR	21:0x49080	0.1	0.0	0.0	1	0.0	0.1	0.1	91	91	112	6	2591	143

Max Plog pages/sec	Max Llog pages/sec	Max Dskflush Time	Avg Dskflush pages/sec	Avg Dirty pages/sec	Blocked Time
461	4007	5	190	7	5

```
informix@cte2:~>
```

Bufferpool Added Capabilities

- **Dynamic Buffer pools have been requested for a while by our users.**
- **We will therefore add the ability to dynamically extend a buffer pool and do so automatically, which will allow us to start with relatively small pools and increase the size of the pools only as needed.**
- **This will improve performance in many cases by getting the memory resources necessary to the database server it needs, if available, to increase application transaction performance much sooner, in near real-time, rather than after the fact, which has been the norm.**

BUFFERPOOL Configuration Parameter Expansion

- **Three goals:**
  - Use a new memory format method of allocating pool memory:
    - Specifying buffer pool size in bytes (Kb, Mb, Gb) units.
  - Marking the pool as extendable.
  - Maintain the legacy format method of **BUFFERPOOL** for current user compatibility:
    - Specifying buffer pool size and its limits in pagesize units.

- **Some BUFFERPOOL arguments are now incompatible with others.**

- **Must follow rules to properly set this parameter:**
  - Order of precedence for all **BUFFERPOOL** definitions:
    - All definitions shall have the same format (all legacy or all memory) or the server will not boot.
    - If an argument's value is not defined, we use the value from the default **BUFFERPOOL** setting.
    - If the default **BUFFERPOOL** setting is not defined, use the internal definition for the format being used. If unable to determine format (ex. **BUFFERPOOL size=2k**), use internal format based on any other **BUFFERPOOL** definitions present. If not, use internal legacy.

Bufferpool Added Capabilities

- **Detection**
  - We want to add more memory/buffers to a buffer pool only when the working set doesn't fit.
  - To determine when to extend the buffer pool, we keep track of the cache hit ratio of each buffer pool by sampling the cache hit ratio once a minute for 5 minutes.
  - If the average cache hit ratio is less than a configured threshold, we extend the buffer pool.

- **Extension**
  - In order to support buffer pool extension, we need to move buffer pools into their own shared memory segments.

- **Shared Memory in Informix can now have 5 possible segments**
  - Resident
  - Virtual
  - Message
  - Bufferpool *** **NEW** ****
  - Extended

Bufferpools

- ***Bufferpool Segments***
  - Buffer pools will no longer be allocated within the resident segment.
  - We will add a new IDS memory segment class that will contain only buffer pool memory structures and adhere to the **RESIDENT** flag.
  - The buffer pool class segment will have virtually no overhead (bitmap or TTree) for maintaining the segment. This will allow for maximum memory usage.

- ***Bufferpool Extending for Legacy Format***
  - The **BUFFERPOOL** configuration parameter's legacy "buffers" format uses these attributes:
    - **buffers** – starting number of buffers
    - **next_buffers** – is the number of buffers to add when the buffer pool is extended and will be doubled every 4th extension *.
    - **max_extends** – maximum number of extensions the buffer pool is allowed

* (See the **onstat –g buf** slide example further below.)

Bufferpools (cont'd)

- ***Bufferpool Extending for Memory Format***
  - The **BUFFERPOOL** configuration parameter's new "Memory" format uses these attributes:
    - **memory** – target amount of memory that a buffer pool can grow to. Values will be rounded up to **32mb** alignments.
    - **start_memory** – starting amount of memory
  - Both **memory** and **start_memory** values can be expressed in a variety of units.
  - For example:

    BUFFERPOOL size=4k,start_memory=32mb,memory=54gb
    BUFFERPOOL size=8k,start_memory=48000,memory=204mb

BUFFERPOOL Configuration Parameter Expansion

- **A BUFFERPOOL definition can NOT include aspects of both formats; either it's expressed in legacy format or memory format.**

- **A BUFFERPOOL definition with mixed formats is rejected and the database server will not start.**

- **The charts on the next several slides describe each argument to the BUFFERPOOL parameter:**
  - **both** = works with legacy & memory format.
  - **legacy & memory** are mutually exclusive.

BUFFERPOOL Argument Compatibility Chart

Argument	Format	Dynamic	Values
size	both	no	Required. Default,2k,4k,….
lrus	both	no	Optional Proposed change internal default from numcpuvps to 4 * physical processors on the box.
lru_min_dirty	both	Yes, auto	Optional Internal default = 50
lru_max_dirty	both	Yes, auto	Optional Internal default = 60
extendable	both	no	Optional 0=FALSE 1=TRUE Determines if bufferpool is extendable or not. Internal default: legacy = 0, memory = 1
cache_hit_ratio	both	Yes	Optional Min=0, Max=100, default = 90 Only applies if bufferpool is extendable

BUFFERPOOL Argument Compatibility Chart (cont'd)

Argument	Forma	Dynamic	Values
buffers	legacy	No	# of buffers Initial # of buffers in 1st segment. Min=MIN_BUFFERS(1000) Max=MAX_BUFFERS(based on virtual memory) Internal default=1000
next_buffers	legacy	Yes, won't impact existing segments or growth projections, only next segment	Requires extendable=1, otherwise, ignored. Min=MIN_BUFFERS(1000) Max=MAX_BUFFERS(based on virtual memory) Internal default=1000 Specifies amount of growth the next bufferpool segment will be.
max_extends	legacy	No	Optional Requires extendable=1, otherwise, ignored. Maximum # of segments the bufferpool can expand to Min=0 Max=MAX_BUFFSEGS (platform specific… 64bit=24, 64bit NT=8, 32bit=16) Internal Default=8

BUFFERPOOL Argument Compatibility Chart (cont'd)

Argument	Format	Dynamic	Values
start_memory	memory	No	Optional for new format determines size of initial bufferpool segment. Min=32Mb Max=Amount of free memory auto=IDS determines starting memory Internal default=Auto
memory	memory	Yes	Required for new format Determine max size bufferpool can grow to Min=32Mb Max=? auto=IDS determines maximum amount of memory to use Internal default=Auto

BUFFERPOOL examples

BUFFERPOOL size=2k,extendable=1,buffers=1000,next_buffers=2000,max_extends=8

- Create a 2k pagesize buffer pool.
- The buffer pool is extendable up to 8 times.
  - The 1st 3 extensions will be of 2000 buffers,
  - Next 4 will be of 4000 buffers and
  - The 8th extension will be of 8000 buffers.

BUFFERPOOL size=10k,start_memory=auto,memory=4gb

- Create a 10k pagesize buffer pool.
- The buffer pool will be extendable and have a max size of 4gb.
- IDS will determine the initial # of buffers and the extension sizes.

- Note: when using legacy format, it is possible to ask for N buffers and get slightly more or fewer depending on memory alignment. We allocate extra memory to allow for alignment and then consume all of the memory allocated.

onstat –p – Monitor your cache hit ratio

```
IBM Informix Dynamic Server Version 12.10.FC3 -- On-Line -- Up 2 days 20:52:24 -- 240192 Kbytes

Profile
dskreads   pagreads   bufreads   %cached dskwrits   pagwrits   bufwrits   %cached
9386       17602      37382022   99.98   56327      243705     1315030    95.72

isamtot    open       start      read       write      rewrite    delete     commit     rollbk
19924297   1404052    1763391    11553575   219313     113026     28859      36847      23

gp_read    gp_write   gp_rewrt   gp_del     gp_alloc   gp_free    gp_curs
201        245939     240450     7          2001       0          18

ovlock     ovuserthread ovbuff     usercpu  syscpu   numckpts   flushes
0          0              0        153.67   42.96    63         181

bufwaits   lokwaits   lockreqs   deadlks    dltouts    ckpwaits   compress   seqscans
21         0          45304484   0          0          78         13109      182106

ixda-RA    idx-RA     da-RA      logrec-RA  RA-pgsused lchwaits
0          1160       0          0          0          1861
```

- **So if the %cached for reads is less that 95% then this is usually a good bet that you are doing too much disk i/o and not enough reading from memory, which does affect performance and this is usually because there are not enough memory buffers to cache all of the data.**

onstat –g seg – Memory Segments in Use

onstat -g seg | Run | Display onstat Options

```
onstat -g seg

IBM Informix Dynamic Server Version 12.10.FC3 -- On-Line -- Up 2 days 20:39:54 -- 240192 Kbytes

Segment Summary:
id         key        addr               size             ovhd     class blkused  blkfree
458759     52864801   44000000           9043968          539144   R     2208     0
491528     52864802   448a0000           103809024        1218120  V     23278    2066
622604     52864806   4aba0000           67108864         1        B     16384    0
688142     52864808   4eba0000           33554432         1        B     8192     0
1179674    52864809   50ba0000           3244032          39384    V     10       782
1212443    5286480a   50eb8000           3244032          39384    V     10       782
1245212    5286480b   511d0000           3244032          39384    V     10       782
1277981    5286480c   514e8000           3244032          39384    V     10       782
1310750    5286480d   51800000           3244032          39384    V     10       782
1343519    5286480e   51b18000           3244032          39384    V     10       782
1376288    5286480f   51e30000           3244032          39384    V     10       782
1409057    52864810   52148000           3244032          39384    V     10       782
1441826    52864811   52460000           3244032          39384    V     111      681
1474595    52864812   52778000           3244032          39384    V     10       782
Total:     -          -                  245956608        -        -     50263    9785

   (* segment locked in memory)
No reserve memory is allocated
```

- As before, the ‘class’ column indicates the type of memory segment, the ‘B’ within the column indicating the new ‘Buffer pool’ segment, and as before: ‘R’ indicating resident and ‘V’ indicating virtual segments.

onstat –g buf – Monitor Buffer Pools – Memory Format

```
Profile

Buffer pool page size: 2048
dskreads   pagreads   bufreads   %cached dskwrits   pagwrits   bufwrits   %cached
9321       17313      34972029   99.97   54808      237719     1127160    95.14
bufwrits_sinceckpt  bufwaits   ovbuff     flushes
34152               18         0          84

Fg Writes     LRU Writes    Avg. LRU Time Chunk Writes  Total Mem
6             0             -nan          28740         64Mb

                                    cache
# extends  max memory  next memory  hit ratio  last
0          512Mb       64Mb         90         15:17:39

Bufferpool Segments
id segment      size     # buffs
0  0x4aba0000   64Mb     25525

----------------------------------

Buffer pool page size: 8192
dskreads   pagreads   bufreads   %cached dskwrits   pagwrits   bufwrits   %cached
20         234        415330     100.00  1459       5836       186287     99.22
bufwrits_sinceckpt  bufwaits   ovbuff     flushes
11806               3          0          7

Fg Writes     LRU Writes    Avg. LRU Time Chunk Writes  Total Mem
0             0             -nan          1459          32Mb

                                    cache
# extends  max memory  next memory  hit ratio  last
0          32Mb        0Mb          90         15:19:28

Bufferpool Segments
id segment      size     # buffs
0  0x4eba0000   32Mb     3966

----------------------------------

Fast Cache Stats
gets       hits       %hits   puts
2914391    2891769    99.22   19871069
```

- **Produces output per bufferpool in use.**
- **Will report either format in use.**

onstat –g buf – Legacy Format

```
Buffer pool page size: 8192
dskreads   pagreads   bufreads   %cached dskwrits   pagwrits   bufwrits   %cached
49584374   205832031  481163349  89.69   17913313   71678428   124805268   85.65

bufwrits_sinceckpt  bufwaits   ovbuff     flushes
1400740             3129838    24630      1919

Fg Writes     LRU Writes    Avg. LRU Time Chunk Writes  Total Mem
24630         15914124      0.000         1973719       863Mb

                                    cache
# extends  max extends  next buffers  hit ratio   last
18          24           16000         90          14:30:25

Bufferpool Segments
id segment      size     # buffs
0  0x4c509000   8Mb      1000
1  0x5315d000   8Mb      1000
2  0x5395f000   8Mb      1000
3  0x54387000   8Mb      1000
4  0x54daf000   15Mb     2000
5  0x55fd4000   15Mb     2000
6  0x5741a000   15Mb     2000
7  0x58419000   15Mb     2000
8  0x59418000   31Mb     4000
9  0x5b858000   31Mb     4000
10 0x5d851000   31Mb     4000
11 0x600d8000   31Mb     4000
12 0x631e3000   63Mb     8000
13 0x67a59000   63Mb     8000
14 0x6c2cf000   63Mb     8000
15 0x777c9000   63Mb     8000
16 0x7b7b6000   127Mb    16000
17 0x84897000   127Mb    16000
18 0x8d978000   127Mb    16000

Fast Cache Stats
gets       hits       %hits   puts
95547659   95334862   99.78   100403152
```

- Really poor read performance here (89.69%) – **ovbuff** and **Fg Writes** have the same number (24630), indicative that you were out of buffers in the bufferpool, in this case, 24,630 times; solution: add buffers to the bufferpool.

VPCLASS

- **This configuration parameter is now expanded to use with cpu class to make the cpu class extendable.**

  VPCLASS *class,aff=range,noage,num=N,autotune,max=N*
  - **autotune** is optional.
    - When set, IDS will determine when to add another VP of ***class*.**
  - **max** is the maximum number of CPU VPs we can extend to.

- **When using autotune, affinity will not be used for any cpu vps added as part of automatic tuning.**

- **This is again for performance reasons, adding an additional cpu vp dynamically thru autotune allows more resource availability to make those transactions complete quicker due to less likelihood of there being not enough cpu to complete a given operation quickly.**
  - Manual intervention is likely to be not as quick in many such cases.

CPU VP's

- **Detection**
  - **CPU VPs** are added whenever the average ready Q count across all the **CPU VPs** for the last minute exceeds **4**.
  - If the average depth of the ready Q is >= **5**, we add another **CPU VP** up to a maximum of the number of physical processors on the box.
  - Monitoring the ready queue **(onstat –g rea,** large number of threads sitting waiting**)**, and the global usage of the CPU virtual processors **(onstat –g glo**, the **eff** column always at 100**)**.

- **Configuration**
  - A new argument to the **VPCLASS** configuration parameter format has been added: **autotune.**
  - This argument will be processed only for the **CPU** and **AIO** vp classes. For more info see **VPCLASS** below.

- **Expansion**
  - Internal procedure to add a **cpu vp**.

  - See Appendix A for Event Alarm relevant to this.

onstat –g rea – Threads Ready to Run

onstat –g rea

IBM Informix Dynamic Server Version 12.10.FC3 -- On-Line -- Up 2 days 23:53:12 -- 240192 Kbytes

Ready threads:

tid	tcb	rstcb	prty	status	vp-class	name
6	536a38	406464	4	ready	3cpu	main_loop()
28	60cfe8	40a124	4	ready	1cpu	onmode_mon
33	672a20	409dc4	2	ready	3cpu	sqlexec

onstat –g cpu

```
onstat -g cpu

IBM Informix Dynamic Server Version 12.10.FC3 -- On-Line -- Up 2 days 23:32:49 -- 240192 Kbytes

Thread CPU Info:
 tid     name              vp       Last Run            CPU Time     #scheds     status
 2       lio vp 0          3lio*    12/10 15:17:41         0.0000           1    IO Idle
 3       pio vp 0          4pio*    12/10 15:17:43         0.1081           2    IO Idle
 4       aio vp 0          5aio*    12/13 14:49:22        30.2746       23066    IO Idle
 5       msc vp 0          6msc*    12/13 14:46:00         0.7297          73    IO Idle
 6       fifo vp 0         7fifo*   12/10 15:17:46         0.1082           2    IO Idle
 7       main_loop()       1cpu     12/13 14:50:27        10.0607      239713    sleeping secs: 1
 8       soctcppoll        8soc*    12/13 14:50:27    257565.6039     1080180    running
 9       soctcplst         1cpu*    12/13 14:50:26         7.1441         663    sleeping forever
 10      soctcplst         1cpu*    12/10 15:17:47         0.0005           9    sleeping forever
 11      soctcplst         1cpu*    12/12 07:50:22         0.3384         153    sleeping forever
 12      flush_sub(0)      1cpu     12/13 14:50:27         3.5955      239570    sleeping secs: 1
 13      flush_sub(1)      1cpu     12/13 14:50:27        14.5295      238556    sleeping secs: 1
 14      kaio              1cpu*    12/13 14:50:27       128.4919      105317    IO Idle
 15      aslogflush        1cpu     12/13 14:50:27         0.7686      238261    sleeping secs: 1
 16      btscanner_0       1cpu     12/13 14:50:27         3.1611       10560    sleeping secs: 16
 17      readahead_0       1cpu     12/13 14:49:02         0.1658         889    cond wait  ReadAhead
 18      auto_tune         1cpu     12/13 14:50:27         2.7852      238198    sleeping secs: 1
 19      onmode_mon        1cpu*    12/13 14:50:27         4.2392      238208    sleeping secs: 1
 20      periodic          1cpu     12/13 14:50:27         5.8948      238192    sleeping secs: 1
 21      memory            1cpu     12/10 15:17:49         0.0372           1    sleeping forever
 176     bf_priosweep()    1cpu     12/13 01:00:45         0.3444         183    cond wait  bp_cond
 184     aio vp 1          9aio*    12/13 09:42:59         2.6009         242    IO Idle
 185     flush_sub(2)      1cpu     12/13 14:50:27         1.2160      238311    sleeping secs: 1
 192     dbScheduler       1cpu     12/13 14:50:03         7.7248       72239    sleeping secs: 276
 193     dbWorker1         1cpu     12/13 14:19:55        12.2121       23221    sleeping forever
 194     dbWorker2         1cpu     12/13 14:19:06        12.9201       17033    sleeping forever
 285     sqlexec           1cpu     12/10 16:41:12         0.0037          29    cond wait  netnorm
 286     sqlexec           1cpu     12/10 16:41:13         0.0054         121    cond wait  netnorm
 287     sqlexec           1cpu     12/10 16:43:58         0.0032          35    cond wait  netnorm
 288     sqlexec           1cpu     12/10 16:41:42         0.0044         121    cond wait  netnorm
 289     sqlexec           1cpu     12/10 16:46:10         0.0096         129    cond wait  netnorm
 447     sqlexec           1cpu     12/11 14:58:22         0.2414          27    cond wait  netnorm
 448     sqlexec           1cpu     12/11 14:48:46         0.0921          89    cond wait  netnorm
 458     smxRecvSnd        1cpu     12/11 14:51:03         0.0858          81    terminated
 463     aio vp 2         10aio*    12/13 01:00:38         0.4769          52    IO Idle
 464     flush_sub(3)      1cpu     12/13 14:50:27         1.7484      153772    sleeping secs: 1
 468     CDRCparse         1cpu     12/11 14:58:24         0.4507          38    cond wait  CDRCparse
 469     CDRGfan           1cpu     12/13 14:50:15         0.4742       11491    sleeping secs: 3
 470     CDRGeval0         1cpu     12/13 14:50:25         0.6912       33997    sleeping secs: 3
 471     CDRGeval1         1cpu     12/13 14:50:25         0.9562       34005    sleeping secs: 3
```

onstat –g glo – Monitor your CPU usage

```
IBM Informix Dynamic Server Version 12.10.FC3 -- On-Line -- Up 2 days 23:31:44 -- 240192 Kbytes

MT global info:
sessions threads  vps      lngspins time
37       91       11       0        257503

          sched calls     thread switches yield 0   yield n   yield forever
total:    14518690        3528477         11204302  2556676   169036
per sec:  0               0               0         0         0

Virtual processor summary:
 class       vps       usercpu   syscpu    total
 cpu         1         144.42    19.89     164.31
 aio         3         0.56      11.11     11.67
 lio         1         0.45      0.45      0.90
 pio         1         0.53      0.55      1.08
 adm         1         3.97      3.61      7.58
 soc         1         11.03     7.64      18.67
 msc         1         0.01      0.00      0.01
 cdrsmi      1         0.07      0.04      0.11
 fifo        1         0.45      0.42      0.87
 total       11        161.49    43.71     205.20

Individual virtual processors:
 vp    pid       class       usercpu   syscpu    total     Thread    Eff
 1     30169     cpu         144.42    19.89     164.31    509.60    32%
 2     30170     adm         3.97      3.61      7.58      0.00      0%
 3     30171     lio         0.45      0.45      0.90      0.90      100%
 4     30172     pio         0.53      0.55      1.08      1.08      100%
 5     30173     aio         0.17      9.62      9.79      30.20     32%
 6     30209     msc         0.01      0.00      0.01      0.72      1%
 7     30210     fifo        0.45      0.42      0.87      0.87      100%
 8     30211     soc         11.03     7.64      18.67     NA        NA
 9     30396     aio         0.22      1.11      1.33      2.53      52%
 10    28871     aio         0.17      0.38      0.55      0.55      100%
 11    28071     cdrsmi      0.07      0.04      0.11      0.24      45%
                 tot         161.49    43.71     205.20
```

AIO VP with AUTOTUNE

- **Use a VPCLASS configuration parameter entry for the AIO virtual processor class to specify an exact number of AIO virtual processors or to enable the database server to add AIO virtual processors as needed.**

- **Again this being done for performance, it is sometimes hard for a DBA to know when he might need an AIO VP and where to look to get this information.**

- **AIO VP's are used on operating systems that do not support KAIO, the database server uses the AIO class of virtual processors to perform database I/O that is not related to physical or logical logging.**
  - The database server uses the CPU class to perform KAIO for database I/O when KAIO is available on a platform.
  - If the database server implements KAIO, a KAIO thread performs all I/O to raw disk space, including I/O to the physical and logical logs.

AUTO_TUNE_SERVER_SIZE

- **Configuration parameter used to buffer pool size when automatic tuning is not possible.**

- **When BUFFERPOOL:memory is set to auto, we set memory using AUTO_TUNE_SERVER_SIZE.**
  - Sets internal structures to assume that's how much memory will get used.
  - Therefore, it is possible to exceed these amounts.

AUTO_TUNE_SERVER_SIZE [OFF | SMALL | MEDIUM | LARGE | XLARGE]

OFF	use 10% available memory
SMALL	use 10% available memory
MEDIUM	use 20% available memory
LARGE	use 33% available memory
XLARGE	use 50% available memory

AUTO_TUNE_SERVER_SIZE

- **This is set only when you install the product and request, as part of the installation process, that a server instance be created:**
  - It sets the sizes of memory and storage spaces to allocate based on the number of expected concurrent users provided by the user during the install.
    - **SMALL** = 1 - 100 users
    - **MEDIUM** = 101 - 500 users
    - **LARGE** = 501 - 1000 users
    - **XLARGE** = more than 1000 users

- **The setting affects the following:**
  - The size of the buffer pool.
  - The maximum size of logical log files (from 200 MB up to 2 GB) before the server stops automatically adding logical logs to improve performance
  - The initial size (from 50 MB to 500MB) of the following created storage spaces, which are created automatically during installation:
    - An extendable plogspace for the physical log
    - A dbspace for the logical log
    - Dbspaces for databases and tables
    - A temporary dbspace
    - An sbspace
    - A temporary sbspace

AUTO_TUNE_SERVER_SIZE

- **This parameter can be used even if you did not create a server during installation, or if you change its value after you initialize the server for the first time.**

- **In this case, the new value affects the size of only the following properties:**
  - The size of the buffer pool, if the **BUFFERPOOL** configuration parameter setting includes the **memory='auto'** option.
  - The maximum size of all logical log files before the server stops automatically adding logical logs to improve performance.

- **Auto-instance-creation with the physical log, logical log and smart large object spaces, and 2 tempspaces all in their own separate dbspaces, along with the separately created rootdbs, will lead to less down time, fewer configuration steps, and increased 'out-of-the-box' experiences for all users, from the very first initial install.**

- **At this point, all they have to do is create databases. But where?**

AUTOLOCATE Configuration Parameter

- **This controls the dbspace location of user defined databases, tables, and indexes and the automatic fragmentation of tables.**

- **If 0 (default) this is disabled:**
  - New databases still default created in rootdbs
  - New table and indexes still default created in the same dbspace as the database

- **Set from 1 to 32, enables automatic location and fragmentation:**
  - Indicates how many round-robin fragments to initially allocate to a table.

- **Stores new databases, tables, and indexes in server defined optimal dbspaces:**
  - By default, all dbspaces are available.
  - Users can control the list of available dbspaces.
  - Fragments new tables by round-robin, where the number of fragments is equal to the value of the **AUTOLOCATE** configuration parameter.
  - Adds more table fragments as the table grows.
  - "**In**" dbspace_name clause overrides

AUTOLOCATE Configuration Parameter

- **If enabled, you can use the autolocate database arguments with the admin() or task() function to**
  - Manage the list of dbspaces for automatic location and fragmentation:
    - The list of available dbspaces is in the **sysautolocate** system catalog table.
  - Disable automatic location and fragmentation for the specified database.

- **You can use the AUTOLOCATE environment option of the SET ENVIRONMENT statement of SQL to enable or disable the value of the AUTOLOCATE configuration parameter for a session.**

- **It is anticipated that the implementation of this feature will lead to fewer out of storage space errors and fewer databases and objects being created in under sized/allocated storage spaces.**

- **Users can still override automatic server based allocation of location by using the "in dbspace" clause of the create database, table and index clauses**

AUTOLOCATE

- **It is anticipated that this will lead to fewer outages caused by having user defined databases and tables and indexes being stored in the rootdbs and causing that space to fill up completely (and stop the instance) or extend/expand unnecessarily beyond its original size.**
  - This has been a long standing user issue and seen in many places.
- **This is a good feature.**
- **Stands alone as a feature. In combination with the implementation of the storage pool it will be very difficult, if configured and administered properly initially, to run out of space.**
- **Best practice. With the storage pool feature implemented.**

SQL Admin API / onmode –wf/wm

- **Turn off autolocation for the stores database:**
  - **execute function task(“autolocate database off”, “stores”);**
- **Turn on autolocation for the stores databases:**
  - **execute function task(“autolocate database add”, “stores”, “datadbs1”);**
  - **execute function task(“autolocate database add”, “stores”, “datadbs1,datadbs2,datadbs3,datadbs4”);**
  - **execute function task(“autolocate database anywhere”, “stores”);**
- **Remove datadbs2 from the list of available dbspaces for the stores database:**
  - **execute function task("autolocate database remove", “stores", "datadbs1");**
- **onmode –wf/wm**

```
id   name                      type   maxlen   units   rsvd   tunable
233  AUTOLOCATE                INT4   12                      *

     min/max : 0,32
     default : 0
     onconfig:
     current : 0
```

Questions

Appendix A – New Event Alarms

Class Id	Internal Name	Event ID	Description
24	ALRMU_85_AUTOTUNE_DAEMON_FAIL	24013	Auto Tune daemon has insufficient resources to perform tuning operations.
Online log: Performance Advisory			
User action: Restart the server with more resources			

Class Id	Internal Name	Event ID	Description
24	ALRMU_85_AUTOTUNE_ADD_CPUVP_FAIL	24014	Auto tuning failed to start a CPU VP
Online log: Performance Advisory			
User action: Restart the server with more resources			

Appendix A – New Event Alarms (cont'd)

Class Id	Internal Name	Event ID	Description
24	ALRMU_24_ADD_BP_FAIL	24015	Auto tuning was unable to extend the buffer pool due to insufficient resources.
Online log: Performance Advisory			
User action: Restart the server with more resources			

Class Id	Internal Name	Event ID	Description
85	ALRMU_85_OVER_LLOG	85001	Auto tuning failed to add another logical log, because adding another log would exceed the maximum log space as defined by configuration parameter AUTO_LLOG.
Online log: Performance Advisory			
User action: Increase the maximum amount of log space by changing AUTO_LLOG configuration parameter.			

Appendix A – New Event Alarms (cont'd)

Class Id	Internal Name	Event ID	Description
85	ALRMU_85_OVER_BPOOL	85002	Auto tuning failed to extend a bufferpool, because the buffer pool would exceed the maximum amount of memory or extensions as defined by configuration parameter BUFFERPOOL.
Online log: Performance Advisory			
User action: Increase the maximum amount defined by the configuration parameter BUFFERPOOL.			

Appendix A – New Event Alarms (cont'd)

Class Id	Internal Name	Event ID	Description
85	ALRMU_85_OVER_CPU	85003	Auto tuning failed to add a CPU VP because another CPU VP would exceed the maximum number specified in the configuration parameter VPCLASS or exceed the number of processors on the computer.
Online log: Performance Advisory			
User action: If there are more processors available on the computer, increase the maximum number of CPU VPs allowed by changing the VPCLASS configuration parameter.			