IBM Information Management

IBM Informix Dynamic Server (IDS) Cheetah v11.10:

RTO_SERVER_RESTART and Non-Blocking Checkpoints

Author: Scott Lashley

February 5, 2007

Informix IDS uses memory (bufferpool[1]) to cache application transaction updates (inserts/updates/deletes). This improves performance dramatically by not having to flush transactions to stable storage (disk) upon commit. The updates are also logically recorded to stable storage (called the logical log or redo log) so that in the event of an unexpected outage, all the transactions which have been committed can be restored. When an unexpected outage does occur, Informix IDS will access the redo log and perform the logical updates to the system in the order they were originally updated to restore the system to a transactionally consistent state at the time the system was interrupted. During normal operation, the transactional updates that are contained within the bufferpool must be flushed to disk periodically to allow Informix IDS to recover from an unexpected outage in a timely manner. If no updates are ever flushed to disk from the bufferpool, then IDS must recover the system starting with the first update ever done. By periodically flushing to disk the updates contained in the bufferpool, IDS can limit the amount of updates required to be replayed during fast recovery.

The points in time which IDS periodically flushes the bufferpool to disk are known as checkpoints. Previous versions of IDS support a blocking checkpoint algorithm that causes the system to block all transactional updates while the bufferpool is flushed to disk. With the introduction of Informix IDS V9, an additional checkpoint algorithm was introduced called Fuzzy Checkpoint. This was introduced as an attempt to limit the amount updates required to be flushed to disk during checkpoint processing. Unfortunately, the algorithm did not alleviate the blocking of transactions entirely during the flush of the bufferpool to disk.

A major problem for many of our customers is how to tune checkpoints. Applications that are sensitive to response time requirements tune LRU flushing to be very aggressive to limit the transactional blocking during a checkpoint. Applications which have a Recovery Time Objective (RTO) policy try to control the frequency of checkpoints to limit the time IDS will take to recover from an unexpected outage. Applications which require both policies find that they work against each. Too frequent checkpoints cause unwanted transaction blocking; not frequent enough checkpoints exposes IDS to a potential long fast recovery time. Tuning a server to meet both policies can be frustrating, especially if the application workload is variable.

For Informix IDS V11.10, a new checkpoint algorithm is being introduced that is virtually non-blocking. Transactional updates are blocked for a very short time (typically, a fraction of 1 second) while a checkpoint point is started. Then, transactions are free to perform updates while the bufferpool containing all the transactional updates is flushed to disk. By not blocking during the bufferpool flush, LRU flushing can be less aggressive which in turn gives more CPU to the application to perform transactions. Also, because checkpoints do not block transactional updates, checkpoints can be done more frequently too help meet RTO policies.

For Informix IDS V11.10, a new ONCONFIG parameter is being introduced, RTO_SERVER_RESTART, which stands for Recovery Time Object Server Restart. This parameter allows the DBA to specify a target amount of time, in seconds, that IDS can utilize

[1] Throughout this document, the term bufferpool refers to 1 or more bufferpools as configured for the system.

during fast recovery to bring the server back online after an unexpected outage. When the ONCONFIG parameter is defined, IDS will control the frequency of checkpoints including automatically adjusting for variable workloads, to make sure IDS can meet the target fast recovery time.

Tuning RTO_SERVER_RESTART

As part of normal transactional updates, IDS transaction management uses the physical log to save off *before images* of pages that are being updated and the logical log to record transactional updates. The physical log restore step of fast recovery restores the database to a physically consistent state prior to the log replay step. The log replay step applies log records to bring the database to a transactionally consistent state at the time of the unexpected outage.

A problem that can occur as part of logical replay is that each of the log records can generate an I/O. If that I/O requires a page to be read from disk, this can dramatically reduce log replay performance. Not knowing how many I/Os might occur during fast recovery can make fast recovery time unpredictable.

In order to make fast recovery more predictable, IDS must control the amount of I/O that will occur during fast recovery. When RTO_SERVER_RESTART is configured to be used, IDS will utilize the physical log to save additional *before images* as part of transaction management with the intent that during fast recovery, no random I/Os will occur because all the pages required by log replay will first be seeded into the bufferpool during physical log restore.

RTO_SERVER_RESTART must be turned on prior to IDS instantiation for it to be effective. Turning on RTO_SERVER_RESTART will have no impact on the current fast recovery, only the next fast recovery, should an unexpected outage occur.

PHYSFILE – Physical log size
For Informix IDS V11.10, the physical log size, ONCONFIG parameter PHYSFILE, is recommended to be 110% of the combined size of all bufferpools. This is to allow fast recovery to utilize all the bufferpool resources for recovering the database as quickly as possible to maintain the RTO_SERVER_RESTART policy. While it is not a requirement for transaction correctness, it provides the optimum resources for peak fast recovery performance whether RTO_SERVER_RESTART is active or not.

If the system has a huge bufferpool where only a portion of the bufferpool is filled with updates, then a smaller physical log size will work just fine. Having a large physical log also will not impact performance. Typically, a physical log size of 1Gb to 4Gb is sufficient depending on the transactional workload and speed of the disks.

Extra physical logging? How will that impact my application?

The new checkpoint algorithm requires more physical log activity. This is because Fuzzy Checkpoint has been deprecated. Applications configured for IDS V7.x should notice little or no change in the rate of physical logging activity. Applications that are tuned for Fuzzy Checkpoint which was introduced in IDS V9.x might experience an increase in physical logging depending upon the type of transactions. When a server is configured with RTO_SERVER_RESTART, there is even more physical logging activity. The increase in physical logging will generate more frequent checkpoints. This can be easily remedied by increasing the physical log size. A new feature for IDS V11.10 is the ability to change the physical log size and/or location using *onparams* without requiring quiescent mode or a system reboot.

In a worse case scenario, as demonstrated by the TPCC benchmark, there was a dramatic increase in physical logging because TPCC transactions update mostly data records and avoid updating indexes. Once the physical log size was increased, there was a negligible decrease in transaction performance (less than 5%) because of the additional physical log activity. This performance decrease was more than compensated for because of the non-blocking checkpoint algorithm and the removal of blocking during checkpoints.

Tuning Checkpoints

In previous versions of IDS, there are 4 conditions that can trigger a checkpoint:

1. Administrative events – archives, adding a dbspace, booting and shutting down the server, onmode -c, etc.
2. 75% full physical log
3. 1 checkpoint in the logical log space – IDS can't overwrite the logical log containing the current checkpoint so it has to trigger a checkpoint prior to moving into that logical log
4. ONCONFIG parameter CKPTINTVL

With the introduction of IDS V11.10, we've introduced a new feature that allows IDS to automatically trigger checkpoints to either maintain RTO_SERVER_RESTART and/or to maintain good checkpoint performance regardless of variation in transaction workload.

CKPTINTVL vs. RTO_SERVER_RESTART

The ONCONFIG parameter CKPTINTVL is used to specify the frequency of checkpoints by allowing the DBA to configure IDS to trigger a checkpoint at specific time intervals. This can be used to control fast recovery time but it does not take into consideration workload activity and therefore is very inflexible often leading to unpredictable results.

When the server is configured with RTO_SERVER_RESTART on, IDS looks at past fast recovery performance along with current transaction activity and automatically triggers a checkpoint in order to maintain both the RTO_SERVER_RESTART policy as well as maintain predictable transaction response times. RTO_SERVER_RESTART and CKPTINTVL are mutually exclusive; only 1 parameter can be active at a time. If the server is configured with RTO_SERVER_RESTART, CKPTINTVL is ignored.

When can transaction blocking occur during a checkpoint?

During checkpoint processing, transactions will continue to consume both physical and logical log resources. The server will not allow transactions to overwrite the physical (known as a physical log overflow) or logical log so update transactions will be blocked until the checkpoint completes. Future checkpoints will take into consideration past checkpoint performance and will trigger checkpoints more frequently to avoid transaction blocking. If checkpoints are triggered because the system is running out of physical or logical log resources[2], consider increasing those resources to reduce checkpoint frequency.

Self Tuning IDS

As mentioned above, IDS V11.10 will automatically trigger checkpoints to help maintain optimal performance. We've added a few more automatic tuning features as well. As with most features, some customers may not wish to take advantage of them. Therefore, we've added new ONCONFIG parameters to control the usage of these features.

Automatic Checkpoints

The ONCONFIG parameter **AUTO_CKPTS** can be used to turn off automatic triggering of checkpoints by ignoring past checkpoint performance. This has the effect of returning the server to triggering checkpoints based only on the 4 events noted above (Tuning Checkpoints). Automatic checkpoints can also be controlled using onmode –wm or –wf. Automatic checkpoints are on by default.

LRU tuning

In pervious versions of IDS, applications that were sensitive to transaction response time required aggressive LRU flushing to reduce the transaction blocking that occurred during a checkpoint. With IDS V11.10, transactions aren't blocked during checkpoint processing and therefore LRU flushing can be relaxed. LRU flushing needs only to deal with page replacement, that is, when a new page that's not in the bufferpool replaces a page that hasn't been accessed in awhile. Should the server detect a condition where frequently accessed read only pages are being replaced, then, the server will automatically make LRU flushing more aggressive to help maintain good performance. A good starting point for setting the LRU flushing parameters are lru_min_dirty=70 and lru_max_dirty=80. Automatic LRU flushing can be turned on/off using the ONCONFIG parameter **AUTO_LRU_TUNING**. It also can be controlled along with adjusting the LRU min and max settings using onmode –wm.

AIO VPs

The use of cooked file chunks is becoming more prevalent. Configuring the server with the correct number of AIO VPs can be difficult. IDS V11.10 will automatically monitor AIO VP performance and when it detects that there are not enough AIO VPs, a new AIO VP will be added. This also includes monitoring the number of cleaner threads and adding more cleaner threads as needed. This feature can be turned on/off using the **AUTO_AIO_VPS** ONCONFIG parameter as well as onmode –wm and –wf.

RAS_PLOG_SPEED

RAS_LLOG_SPEED

These 2 ONCONFIG parameters are used to store the rate at which the physical and logical log can be recovered during fast recovery. RAS_PLOG_SPEED is initially set when the physical log is initialized. RAS_LLOG_SPEED is initialized to 1/8 of RAS_PLOG_SPEED. Each time a fast

[2] See Monitoring Checkpoints below for information on how to determine what event is triggering checkpoints.

recovery occurs, these values are updated to reflect what the real recovery speed is. The units are pages per second.

Why ONCONFIG parameters?
By making these ONCONFIG parameters, it allows our customers that embed IDS into their applications to provide pre-computed values so no tuning is required to get optimal performance. DBAs should never change these values unless directed by IBM Technical Support.

Additional Tuning Recommendations

The following is a list of ONCONFIG parameters that might require additional tuning with the introduction of IDS V11.10.

PHYSBUFF
The default value for the physical log buffer size should be 128Kb. When RTO_SERVER_RESTART is enabled, the default size is 512Kb. If you decide to use a smaller value, a message will be generated indicating that optimal performance may not be attained. Using a physical log buffer smaller than the default size will only impact performance, not transaction integrity.

LOGBUFF
The default value for the logical log buffer size is 64Kb. If you decide to use a smaller value, a message will be generated indicating that optimal performance may not be attained. Using a logical log buffer smaller than 64Kb will only impact performance, not transaction integrity. If the database or application is defined to use buffered logging, increasing the LOGBUFF size beyond 64Kb will improve performance.

LOGFILES & LOGSIZE
It is difficult to determine the amount of log space (LOGFILES * LOGSIZE) that is optimal because it depends on the application requirements and usage. The amount of log space really depends on a few factors:

- Amount of update activity – the more update activity, the more log space that's required.
- Recovery Point Objective (RTO) – in the case of a catastrophic event, how much data loss can be tolerated? By doing more frequent log backups, you reduce your risk of transaction loss.
- Enterprise and HDR replication – both of these replication services can influence the number and size of log files. If your system uses either or both of these replication services, please refer those sections in the manual on sizing the logical log.

Here are a few guidelines:

- It is easier to manage fewer larger log files then to manage lots of smaller log files.
- Having too much log space will not impact performance but not having enough log files and log space can impact performance by triggering frequent checkpoints.
- BlobSpace blobs are not logged but they are included on the log backup in which the blob was created. This means that blobs are not freed until the log in which they were created in are backed up. Therefore, if blobspace blobs are frequently updated, it may require more frequent log backups to acquire more free space within a blobspace.

- A good starting point for applications which generate a small amount of log data is 10 log files of 10Mb each. A good starting point for applications which generate a large amount of log data is 10 log files with 100Mb each.
- In the past, users would try to configure the size of a log file to meet the RPO policy. During steady state, given a certain transaction rate, the log file would fill and generate an automatic log backup. If there is no RPO policy that needs to be maintained, this method works well. But, if a RPO policy is required, a better method is to use the DBCron facility (also introduced in IDS V11.10) to insert a task that executes at the desired frequency to maintain the policy. With DBCron, you can have it automatically back up log files at certain times within the daily cycle. Should the log space fill prior to the logs being backed up and recycled, you can backup the logs, you can add an additional log file to allow transaction processing to continue or you can add a DBCron task to detect this situation and perform either operation automatically.

Additional log files can always be added at any time and the server will automatically add log files when required for transaction consistency, such as with long transactions which can consume large amounts of log space.

Performance Warnings

IDS V11.10 attempts to look at the server configuration and transaction workload and determine if the system is optimally tuned. When it detects that the server is not configured optimally, it will generate a performance warning that is put into the message log (**MSGPATH**). The warnings typically contain a suggesting configuration change that will improve the server's performance. Below are some examples of performance warnings and what they mean.

```
Performance Warning: The physical log size is smaller than the recommended size for a
server configured with RTO_SERVER_RESTART.
Results: Fast recovery performance may not be optimal.
Action: For best fast recovery performance when RTO_SERVER_RESTART is enabled,
increase the physical log size to at least %ld Kb. For servers
configured with a large bufferpool, this may not be necessary. Refer
to the IBM Informix IDS Administration Guide for more information.
```

Typically, the physical log should be 110% of the total size of the combined bufferpools. This is to make sure fast recovery does little or no I/O during roll forward. The actual size that is needed could be quite a bit less than the recommended, especially for systems configured with large bufferpools. This document contains some suggestions on sizing the physical log.

```
Assert Warning: Boot Time '%d' is 50 percent more than RTO_SERVER_RESTART %d
Results: Server initialization is taking a long time and IDS is unable to meet the
RTO_SERVER_RESTART policy. Increase RTO_SERVER_RESTART to at least %d seconds.
Action: Disabling RTO_SERVER_RESTART.
```

This assertion occurs when a normal restart of the server has occurred and for whatever reason, it takes so long for the server to reboot that its not possible to maintain the RTO_SERVER_RESTART policy.

```
Assert Warning: Physical Log is running out of room.
Results: Blocking transactions until checkpoint is complete.
```

```
Action: Increase physical log size.
```

This assertion can occur when a checkpoint is being processed and the physical log is depleted to a critical point where further transactions would cause the physical log to overflow.

```
Assert Warning: Logical log is running out of room.
Results: Blocking transactions until checkpoint is complete.
Action: Increase logical log size.
```

This assertion can occur when a checkpoint is being processed and the logical log is depleted to a critical point where further transactions would cause the logical log to overflow.

```
Assert Warning: Long transactions are triggering blocking checkpoints.
Results: Blocking transactions until checkpoint is complete.
Action: Increase logical log size.
```

Long transactions are triggering frequent checkpoints and transaction blocking must be done to protect critical logical log resources to maintain transactional consistency.

```
Assert Warning: The physical log is too small to accommodate the time it takes to
flush the bufferpool.
Results: Transactions may block during checkpoints.
Action: Increase the size of the physical log to at least %d Kb.
```

IDS started checkpoint processing but can’t flush the bufferpool in a timely manner because transactions are consuming the remaining 25% of the physical log. Increasing the size of the physical log could help remedy this problem.

```
Assert Warning: The logical log is too small to accommodate the time it takes to flush
the bufferpool.
Results: Transactions may block during checkpoints.
Action: Increate the size of the logical log space to at least %d Kb.
```

IDS started checkpoint processing but can’t flush the bufferpool in a timely manner because transactions are consuming the remaining logical log resources. Increasing the size of the logical log space could help remedy this problem.

```
Assert Warning: The physical log is too small for automatic checkpoints.
Results: Automatic checkpoints are disabled.
Action: Increase the physical log size to at least %d Kb.
```

If the server can generate physical log activity faster than automatic checkpoints can handle, then we have to turn off automatic checkpoints. If the physical log < 10Mb (10000Kb) or the server can generate physical log activity so fast that a checkpoint would be generated approximately every 35 seconds, then automatic checkpoints are turned off. This can often occur when the IDS server uses the default settings in the ONCONFIG.std file with no regard for tuning the server. Increasing the physical log size could remedy this problem.

```
Assert Warning: The logical log is too small for automatic checkpoints.
```

```
Results: Automatic checkpoints are disabled.
Action: Increase the logical log space to at least %d Kb.
```

If the server can generate logical log activity faster than automatic checkpoints can handle, then we have to turn off automatic checkpoints. If the logical log < 20Mb (20000Kb) or the server can generate logical log activity so fast that a checkpoint would be generated approximately every 35 seconds, the automatic checkpoints are turned off. This can often occur when the IDS server uses the default settings in the ONCONFIG.std file with no regard for tuning the server. Increasing the logical log space could remedy this problem.

Monitoring Checkpoints

The server keeps track of checkpoint history for the previous 20 checkpoints.

```
onstat –g ckp
IBM Informix Dynamic Server Version 11.100.F       -- On-Line -- Up 00:01:20 -- 299368 Kbytes

Auto Checkpoints=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   163    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      13:188      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
```

Some of the following messages might also be displayed along with the above information.

```
Physical log is too small for bufferpool size. System performance may be less than optimal.
Increase physical log size to at least %ldKb

Physical log is too small for optimal performance.
Increase the physical log size to at least $ldKb.

Logical log space is too small for optimal performance.
Increase the total size of the logical log space to at least %ld Kb.

Transaction blocking has taken place. The physical log is too small.
Please increase the size of the physical log to %ldKb

Transaction blocking has taken place. The logical log space is too small.
Please increase the size of the logical log space to %ldKb
```

Automatic Checkpoints (On/Off)	Indicates if auto checkpoints is on or off
RTO_SERVER_RESTART=## seconds	Indicates RTO policy
Estimated recovery time ## seconds	Indicates estimated recovery time should a server crash occur; This value will only display if RTO_SERVER_RESTART is active.

There are various conditions that can occur where the server is not configured optimally. In this example, two are shown which indicate that the physical log is not large enough and could result in transaction blocking or sub-optimal fast recovery performance. Each message indicates the suggested change to the configuration to improve performance.

Ledger

Interval	Checkpoint interval
Clock Time	Wall clock time when checkpoint occurred
Trigger	Event that triggered checkpoint; events include Admin, Startup, CKPTINTVL, LongTX, Recovery, Backup, Plog, Llog, Misc, RTO, CDR, Pload, Conv/Rev,Reorg, HDR, User, Lightscan
LSN	Logical log position where checkpoint is recorded
Total Time	Total checkpoint duration from request time to checkpoint completion
Flush Time	Time to flush bufferpools
Block Time	Transaction blocking time
# Waits	Number of threads that blocked waiting for checkpoint
Wait Time	Average time thread waited for checkpoint
Long Time	Longest amount of time thread waited for checkpoint
# Dirty Buffers	Number of dirty buffers flushed
Dskflu/Sec	Number of buffers flushed per sec
Plog Total Pages	Total number of pages physically logged during checkpoint interval
Plog Avg/Sec	Average rate of physical log activity during checkpoint interval
Llog Total Pages	Total number of pages logically logged during checkpoint interval
Llog Avg/Sec	Average rate of logical log activity during checkpoint interval
*	Indicates checkpoint requested was a transaction blocking checkpoint

All this information is also located in the following sysmaster tables:
sysckpthistory

```
    create table sysckpthistory
    (
        intvl            int,        {checkpoint interval           }
        type             char(12),   {checkpoint type               }
        caller           char(10),   {caller                        }
        clock_time       int,        {time of day of ckpt           }
        crit_time        float,      {time spent in wait4critex     }
        flush_time       float,      {time spent flushing pages to disk }
        cp_time          float,      {time from cpkt_pending to done }
        n_dirty_buffs    int,        {number of dirty buffers       }
        plogs_per_sec    int,        {avg # pages plogged           }
        llogs_per_sec    int,        {avg # pages logged            }
        dskflush_per_sec int,        {avg # pages dskflushed        }
        ckpt_logid       int,        {LSN of ckpt                   }
        ckpt_logpos      int,        {LSN of ckpt                   }
        physused         int,        {total pages plogged in ckpt   }
        logused          int,        {total pages llogged in ckpt   }
        n_crit_waits     int,        { # of crit section waiters    }
        tot_crit_wait    float,      {total time spent waiting for crit }
        longest_crit_wait float,     {longest crit wait             }
        block_time       float       {blocked time                  }
    );
```

This is a view on top of sysshmhdr.
sysckptinfo

```
            ckpt_status          0
            plogs_per_S          290
            llogs_per_S          669
            dskF_per_S           4
            longest_dskF         0
            dirty_pgs_S          0
            sug_plog_sz          0
            sug_llog_sz          0
            ras_plog_sp          25000
            ras_llog_sp          398
            boottime             6
            auto_ckpts           1
            auto_lru             1
            cur_intvl            2
```

Monitoring I/O activity

With the introduction of interval checkpoints, we've also introduce a high speed clock mechanism that allows us to gather all those nifty statistics cheaply. Onstat –g iof was modified to show the running average that I/O's are taking to each device.

onstat –g iof

```
AIO global files:
gfd  pathname        bytes read   page reads   bytes write   page writes   io/s
3    /dev/sdb5       317440       155          18432         9             570.8
     op type         count        avg. time
     seeks           0            N/A
     reads           0            N/A
     writes          0            N/A
     kaio_reads      27           0.0023
     kaio_writes     9            0.0003

4    /work/chunk     4147200      2025         177547264     86693         617.4
```

```
op type       count     avg. time
seeks         0         N/A
reads         2025      0.0001
writes        1369      0.0040
kaio_reads    0         N/A
kaio_writes   0         N/A
```

Raw devices use KAIO. Cooked devices do not.