PUG Challenge EMEA 2018

Checkpoints – What You Need to Know

Presented by: Dan Foreman
Dan Foreman

• Progress User since 1984
• Author of several Progress related Publications
  – *Progress Performance Tuning Guide*
  – *Progress Database Administration Guide*
  – *Progress System Tables*
• Author of several Progress DBA Tools
  – ProMonitor
  – Pro Dump&Load
  – Balanced Benchmark
• Getting closer to retirement!
Dan Foreman – Part 2

- Interests
  - Basketball
  - Cycling (made in the UK)
  - Backcountry skiing, also called randonee or ski mountaineering
  - Japan (日本)
  - World Travel
Cairo Egypt – 2 days ago
Audience Survey - Demographics

• How many have used Progress for less than one year?
• How many are working for a company that has used Progress for less than one year?
Audience Survey - Technical

• Largest Single Progress DB
• Highest Progress Version
• Lowest Progress Version
• Are you using any of these Progress products?
  – Auditing
  – Multi-Tenancy
  – OpenEdge Replication
  – TDE
  – Table Partitioning
Progress History Test – Unix Platforms

- Plexus
- Arete / Arix
- Convergent Technology
- Fortune
- Sanyo Icon
- DEC (VMS & Ultrix)
- NCR
- AT&T (Unix PC)
- Harris
- ICL
- NEC
- Olivetti
- Siemens Nixdorf
- Prime
- Pyramid
- Sequent
- Sun
- Unisys
- Honeywell
- Silicon Graphics
- Wicat
- Apollo
- MIPS
Obsolete OS formerly supported by Progress:

- BSD
- Ultrix
- BTOS/CTOS
- Xenix
- OS/1
- Netware
- OS/400
- Apple/UX
- SunOS
- MS-DOS
- HP9000
What Happens When a Client Changes the DB

• A buffer (or block) is locked so no one else can change it
• A copy of database block is read from disk into the buffer pool in shared memory (but not if it’s already there)
• The Record is updated within the buffer
  – possibly multiple blocks in multiple buffers
• The Buffer is marked “dirty” (modified)
• The Change is made in memory only
  – It must be written to disk … eventually, not now
• Buffer lock is released
• Now someone else can change it
When are modified blocks written to disk?

• Short Answer: as infrequently as possible for good performance

• If they are never written:
  – To recover after a crash, it would be necessary to redo all changes since database was started
  – The BI Log would grow during the whole time
  – We would run out of buffers in the buffer pool for changed blocks
When are modified blocks written to disk?

- **Writes occur:**
  - If space is required in the buffer pool to bring in a new block
  - When there are many dirty blocks
  - When there is nothing else to do
  - And as additional changes are being made; This is called “Checkpointing”
The BI File (or Log) is composed of a “ring” (a double linked list) of Clusters.

When the BI File grows it grows in Cluster sized units.

A Cluster made of BI blocks (-biblocksize).

Four Clusters (the minimum) are allocated initially.

Clusters are reused when not needed (more about this on the next slide).

Maximum Cluster Size is 256mb.
Cluster Reuse Rules

• When a BI Cluster is filled up:
  – Look at the next Cluster in the ring to see if it can be reused
  – There might be many eligible Clusters in the ring but only the next adjacent Cluster can be examined

• A Cluster can be reused if:
  – No Notes describing changes by open transactions are in the Cluster, i.e. the trx must be committed or rolled back
  – The DB blocks changed by the BI Notes in the Cluster have been written from memory to disk

• Cluster reuse helps keep the BI file as small as possible

• In V9 and older, all transactions in that Cluster had to be committed or rolled back for a minimum of 60 seconds before the Cluster could be reused
What goes into the BI File?

• A BI log record (called a “Note”) is generated for every physical change to the database
• Each Note describes exactly one change to one database block
  – Exceptions - There are log records that only describe changes to memory-resident data structures like the transaction table
• Some activities may require more than one DB change
  – Adding an index entry may cause an index split
  – Adding a record to the DB may require the DB to be extended
• Notes are written in the same order that the DB changes are executed
• Notes generated by concurrent transactions are mixed together
Checkpoint – Definition & Background

• There is a bunch of updated data stored in the DB Buffer Cache, i.e. the “memory resident database state”
• That data needs to get written to disk periodically
• Checkpoints give us the opportunity to perform that task
• Gus Definition:  
  A checkpoint is a process for making what is on disk consistent with the changed or updated database parts that are present only in memory.  
  It is a process, not an event.
  Dan calls it an event or point-in-time just to annoy Gus. And also because some potentially bad things happen at particular points in the Checkpoint process
The Checkpoint Process

• Beginning
• Middle
• End

For more info on CheckPoint internals info please refer to:
pugchallenge.org/downloads2013/224_bi_checkpoints_crashes_v03.pdf
**Checkpoint - Beginning**

- BI buffers flushed
- All dirty (modified) blocks in –B/-B2 placed on Checkpoint Queue
- Next BI Cluster opened
  - Look for a reusable Cluster
  - If no reusable Clusters are available a new one needs to be created

**Checkpoint Timeline**

![Checkpoint Timeline Diagram](image)

- Cluster 1
- Cluster 2
- Cluster 3
Brief Tangent – BI Cluster Basics

• The BI file on a running DB can have a minimum of 4 Clusters (a truncated BI has zero Clusters)
• The BI Clusters are maintained as a ‘ring’, i.e. a double-linked list
• Minimum Cluster size: 16k – Don’t use it unless (maybe) you are forced to use a Workgroup DB License
• Maximum Cluster size: 256mb – Use with Caution!!
Asynchronous Page Writers take blocks off the Checkpoint Queue and write them to disk.

APW’s pace themselves, i.e. attempt to spread out the writes to avoid I/O spikes
• As the Cluster approaches full, all blocks from Checkpoint Queue “should have” been written to disk
• But things don’t always go according to plan
Checkpoint Frequency

- Controlled by:
  - The BI cluster size
  - Application's write activity & behavior
  - The speed of the storage system
- Generally the longer the better; 60 seconds or longer is a good goal
- Unfortunately with the maturation of SSD & Flash Storage, in some cases Checkpoint frequencies can be in the single digit second range even with the maximum BI Cluster Size
06/09/15  Status: BI Log  04:19:08

Before-image cluster age time:  0 seconds
Before-image block size:  8192 bytes
Before-image cluster size:  32768 kb (33554432 bytes)
Number of before-image extents:  1
Before-image log size (kb):  131192

Bytes free in current cluster:  11130610 (34 %)

Last checkpoint was at:  06/09/15 04:19
Number of BI buffers:  20
Full buffers:  19

Hint: when the % counts down to 0, refresh the screen repeatedly & quickly and if there is a pause between 0% and 100%, that pause is also being experienced by all Clients
How to Monitor Checkpoint History

- `promon` has some “recent” Checkpoint history
- Prior to V11.7 only the last 8 Checkpoints are visible
- In V11.7, 32 Checkpoints are visible
  - The 32 can be modified with `-numcheckpointstats` DB startup parameter
Checkpoint – The Bad Stuff

• A Summary of the bad things
  – Buffers Flushed
  – Cluster Formatting
  – OS sync Call (V9 and older)
  – OS fdatasync Call (V10 and above)
  – Other Stuff

• These things occur at the point when a Cluster is filled up
Checkpoint Problem #1 - Buffers Flushed

- Modified DB Buffers (in –B/-B2 Cache) need to be written to the DB
- Those writes are synchronous!
- **ALL** transaction activity is FROZEN until all of the modified buffers have been written
- Solutions:
  - APWs – asynchronously write the modified buffers during the ‘middle’ phase of the Checkpoint
  - A BI Cluster size large enough to give the APWs time to do their job
    - 32-256mb on fast modern hardware
    - Generally 30-60 seconds between Checkpoints is good
    - The default BI Cluster size is 512K !!
## Checkpoint Problem #1 - Buffers Flushed

### Promon > Activity

Activity - Sampled at 06/09/15 04:32 for 0:14:02.

<table>
<thead>
<tr>
<th>Event</th>
<th>Total</th>
<th>Per Sec</th>
<th>Event</th>
<th>Total</th>
<th>Per Sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commits</td>
<td>294637</td>
<td>349.9</td>
<td>Undos</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>Record Updates</td>
<td>0</td>
<td>0.0</td>
<td>Record Reads</td>
<td>375</td>
<td>0.4</td>
</tr>
<tr>
<td>Record Creates</td>
<td>294638</td>
<td>349.9</td>
<td>Record Deletes</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>DB Writes</td>
<td>6408</td>
<td>7.6</td>
<td>DB Reads</td>
<td>161</td>
<td>0.2</td>
</tr>
<tr>
<td>BI Writes</td>
<td>19144</td>
<td>22.7</td>
<td>BI Reads</td>
<td>18</td>
<td>0.0</td>
</tr>
<tr>
<td>AI Writes</td>
<td>0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Record Locks</td>
<td>1178567</td>
<td>1399.7</td>
<td>Record Waits</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>Checkpoints</td>
<td>4</td>
<td>0.0</td>
<td>Buffs Flushed</td>
<td>3025</td>
<td>3.6</td>
</tr>
</tbody>
</table>

Rec Lock Waits 0 %  BI Buf Waits 1 %  AI Buf Waits 0 %  Writes by APW 0 %  Writes by BIW 0 %  Writes by AIW 0 %  Buffer Hits 100 %  Primary Hits 100 %  Alternate Hits 0 %  DB Size 40 MB  BI Size 128 MB  AI Size 0 K  FR chain 59 blocks  RM chain 3 blocks  Shared Memory 17669K  Segments 1
**Checkpoint Problem #1 - Buffers Flushed**

Promon > R&D > Other > Checkpoints

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Len</th>
<th>Freq</th>
<th>Dirty</th>
<th>CPT Q</th>
<th>Scan</th>
<th>APW Q</th>
<th>Flashes</th>
<th>Duration</th>
<th>Sync Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>12:27:38</td>
<td>148</td>
<td>0</td>
<td>10658</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2.51</td>
<td>0.00</td>
</tr>
<tr>
<td>4</td>
<td>12:22:22</td>
<td>316</td>
<td>316</td>
<td>11651</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4854</td>
<td>3.61</td>
<td>0.00</td>
</tr>
<tr>
<td>3</td>
<td>12:15:04</td>
<td>438</td>
<td>438</td>
<td>7647</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6797</td>
<td>0.40</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>12:00:34</td>
<td>870</td>
<td>870</td>
<td>1021</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>850</td>
<td>0.20</td>
<td>0.00</td>
</tr>
<tr>
<td>1</td>
<td>12:00:03</td>
<td>31</td>
<td>31</td>
<td>171</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>171</td>
<td>0.15</td>
<td>0.00</td>
</tr>
<tr>
<td>0</td>
<td>11:57:44</td>
<td>139</td>
<td>139</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Checkpoint Problem #2 – BI Cluster Formatting

• If there are no reusable BI Cluster available, create a new Cluster by allocating space and formatting that space
• Fixed size extents take care of the space allocation
• Fixed size extents DO NOT prevent the formatting
• All transaction activity is FROZEN until the formatting is complete
• Analogy: unformatted diskettes
• Solution:
  – Don’t truncate the BI File…unless necessary
  – Preformat the BI File with `proutil bigrow`
    • Make the script smart, `bigrow appends` to any existing BI file
Checkpoint Problem #3 – sync Call

• In older versions of Progress, a sync call is issued to force the dirty buffers in the OS Cache to disk
• Sync is a synchronous event
• All transaction activity is FROZEN until the sync call completes
• Old Solutions:
  – -directio
  – Tune the sync daemon (60 seconds the common default)
  – Reduce the OS Buffer Cache – RARE!
  – Monitor sync call duration
  – UPGRADE!
Checkpoint Problem #3 – -directio

- Changes database writes to be synchronous, like BI writes
- No OS buffering is involved, so *sync* is no longer needed
- Added in V6
- Originally only worked with DG (Data General) & Sequent
- Starting in V8 applies to all platforms but the documentation was not updated to reflect that change
- Bad news on:
  - Windows
  - Linux
At least it was in some benchmarks a few years ago
Checkpoint Problem #4 – fdatasync Call

- The `sync` call is evil because it flushes the entire OS buffer cache to disk
- In V10 Progress replaced `sync` with `fdatasync`
- The ‘f’ is File…`fdatasync` forces the writes for specific files
- The idea is that the amount I/O required to write dirty DB extents should be less than writing the entire OS buffer cache
- But not always!
- My peers claim that `fdatasync` has made –`directio` obsolete….they are wrong
Checkpoint Problem #4 – fdatasync Call

Promon > R&D > Other > Checkpoints

<table>
<thead>
<tr>
<th>Ckpt No.</th>
<th>Time</th>
<th>Len</th>
<th>Freq</th>
<th>Dirty</th>
<th>CPT Q</th>
<th>Scan</th>
<th>APW Q</th>
<th>Flushes</th>
<th>Duration</th>
<th>Sync Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>968</td>
<td>00:59:07</td>
<td>84</td>
<td>0</td>
<td>40285</td>
<td>30641</td>
<td>377</td>
<td>299</td>
<td>0</td>
<td>4.93</td>
<td>3.48</td>
</tr>
<tr>
<td>967</td>
<td>00:58:32</td>
<td>34</td>
<td>35</td>
<td>41689</td>
<td>33957</td>
<td>245</td>
<td>84</td>
<td>0</td>
<td>5.40</td>
<td>3.74</td>
</tr>
<tr>
<td>966</td>
<td>00:58:03</td>
<td>28</td>
<td>29</td>
<td>41606</td>
<td>33822</td>
<td>343</td>
<td>88</td>
<td>0</td>
<td>6.19</td>
<td>4.27</td>
</tr>
<tr>
<td>965</td>
<td>00:57:31</td>
<td>32</td>
<td>32</td>
<td>44671</td>
<td>36390</td>
<td>320</td>
<td>89</td>
<td>593</td>
<td>6.18</td>
<td>4.99</td>
</tr>
<tr>
<td>964</td>
<td>00:57:03</td>
<td>27</td>
<td>28</td>
<td>47257</td>
<td>39790</td>
<td>182</td>
<td>149</td>
<td>0</td>
<td>6.11</td>
<td>4.72</td>
</tr>
<tr>
<td>963</td>
<td>00:56:34</td>
<td>28</td>
<td>29</td>
<td>49035</td>
<td>41428</td>
<td>439</td>
<td>49</td>
<td>0</td>
<td>4.68</td>
<td>3.09</td>
</tr>
<tr>
<td>962</td>
<td>00:56:07</td>
<td>26</td>
<td>27</td>
<td>31657</td>
<td>24269</td>
<td>217</td>
<td>52</td>
<td>0</td>
<td>3.03</td>
<td>1.00</td>
</tr>
<tr>
<td>961</td>
<td>00:54:35</td>
<td>91</td>
<td>92</td>
<td>20442</td>
<td>12729</td>
<td>650</td>
<td>78</td>
<td>0</td>
<td>2.94</td>
<td>1.08</td>
</tr>
</tbody>
</table>
Checkpoint Problem #5 – Other Stuff

- All dirty –B/-B2 Buffers must scanned and placed on the Checkpoint Queue, i.e. the Checkpoint Queue has to be rebuilt…to do that we must scan the **entire** –B/-B2
- In the new BI Cluster (even one that is already formatted) the Cluster header has to be initialized (one synchronous write)
- Active slot(SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS???) in the Transaction Table (in shared memory) has to be written out
- Until V11.6 and V11.7 there was no way to see the cost of these activities
**Viola!**

- One of the best & most useful screens in *promon*

<table>
<thead>
<tr>
<th>No. Time</th>
<th>Len</th>
<th>Freq</th>
<th>Dirty</th>
<th>CPT Q</th>
<th>Scan</th>
<th>APW Q</th>
<th>DB Writes</th>
<th>Bi Writes</th>
<th>Duration</th>
<th>Sync Time</th>
<th>DB Write</th>
<th>Bi Write</th>
<th>NumChkpt</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 12:57:06</td>
<td>2</td>
<td>0</td>
<td>2088</td>
<td>1063</td>
<td>1</td>
<td>250</td>
<td>0</td>
<td>5</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1315</td>
</tr>
<tr>
<td>5 12:57:00</td>
<td>6</td>
<td>6</td>
<td>2099</td>
<td>2001</td>
<td>7</td>
<td>97</td>
<td>1</td>
<td>31</td>
<td>0.01</td>
<td>0.00</td>
<td>0.00</td>
<td>0.01</td>
<td>2098</td>
</tr>
<tr>
<td>4 12:56:56</td>
<td>4</td>
<td>4</td>
<td>1907</td>
<td>1845</td>
<td>6</td>
<td>62</td>
<td>0</td>
<td>32</td>
<td>0.03</td>
<td>0.02</td>
<td>0.00</td>
<td>0.01</td>
<td>1907</td>
</tr>
<tr>
<td>3 12:56:53</td>
<td>3</td>
<td>3</td>
<td>1267</td>
<td>1267</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.01</td>
<td>0.01</td>
<td>0.00</td>
<td>0.00</td>
<td>1267</td>
</tr>
<tr>
<td>2 12:56:49</td>
<td>4</td>
<td>4</td>
<td>831</td>
<td>831</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>87</td>
<td>0.02</td>
<td>0.01</td>
<td>0.00</td>
<td>0.02</td>
<td>831</td>
</tr>
<tr>
<td>1 12:55:43</td>
<td>66</td>
<td>66</td>
<td>1746</td>
<td>1746</td>
<td>261</td>
<td>0</td>
<td>0</td>
<td>73</td>
<td>0.01</td>
<td>0.00</td>
<td>0.00</td>
<td>0.01</td>
<td>1746</td>
</tr>
<tr>
<td>0 12:54:16</td>
<td>43</td>
<td>87</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
</tr>
</tbody>
</table>
### Checkpoints

<table>
<thead>
<tr>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CkptNo</td>
<td>Unique number assigned to each Checkpoint. Resets when the DB is restarted.</td>
</tr>
<tr>
<td>Time</td>
<td>Starting time of the Checkpoint.</td>
</tr>
<tr>
<td>Len</td>
<td>The length of &quot;execution&quot; time since the previous Checkpoint. The Checkpoint is not considered completed until all dirty buffers on the checkpoint queue have been written out. This is either done over time by the APWs between checkpoints or is forced at the next checkpoint (DB Writes). Pre V10.1B, the Len column reported the elapsed time between Checkpoints. From this you could not tell how long you were actually spending executing a Checkpoint nor how long you were processing between Checkpoints.</td>
</tr>
<tr>
<td>Freq</td>
<td>The Freq column is amount of time between the beginnings of two Checkpoints. (This is the &quot;old&quot; Len column value before Freq was added). The length of time it took to actually execute the Checkpoint can be measured by ( Freq - Len ).</td>
</tr>
<tr>
<td>Dirty</td>
<td>Number of modified blocks in -B at the opening of a new Cluster. These blocks must all be written before the new Cluster is closed (next CP). If they were not they would all be written at the end of the CP and counted as 'Flushed'.</td>
</tr>
<tr>
<td>Label</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CPT Q</td>
<td>Checkpoint Queue. Number of blocks written from the Checkpoint queue (typically) by APWs.</td>
</tr>
<tr>
<td>Scan</td>
<td>Number of blocks written by APWs during the scan cycle (APWs that had nothing else to do; i.e. the CPT Q and APW Q were both empty).</td>
</tr>
<tr>
<td>APW Q</td>
<td>Number of blocks written from the APW queue and replaced on the LRU chain by APWs.</td>
</tr>
<tr>
<td>Flashes or</td>
<td>Number of modified blocks left when the BI cluster filled (Checkpoint event) that had to be written all at once and written synchronously. This number</td>
</tr>
<tr>
<td>DB Writes</td>
<td>should be 0 or a very low number. Renamed to ‘DB Writes in V11.6.</td>
</tr>
<tr>
<td>BI Writes</td>
<td>V11.6.0. The total number of BI Buffers written to disk from the -bibufs at the end of the Checkpoint.</td>
</tr>
<tr>
<td>Label</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Duration</td>
<td>V10.2B SP5. The time required to complete all the housekeeping activities related to the Checkpoint. Duration consists of several distinct activities which are outlined in detail on the next page.</td>
</tr>
<tr>
<td>Sync Time</td>
<td>V10.2B SP5. The time required to perform just the file system synchronization portion of the Checkpoint. This equates to the fdatasync() system call on *ix and FlushFileBuffers() on Windows. “Sync Time” is included in ‘Duration’. AI and BI files are not included unless running with -i or -r.</td>
</tr>
<tr>
<td>DB Write</td>
<td>V11.6.0. The time in seconds used to scan and write the buffers from the database buffer cache (-B and -B2). A large buffer cache can make this value high.</td>
</tr>
<tr>
<td>BI Write</td>
<td>V11.6.0. The time in seconds needed to write the buffers from the BI Buffers (-bibufs) memory.</td>
</tr>
<tr>
<td>NumChkpt</td>
<td>V11.6.0. The number of DB Buffers (from -B and -B2) checkpointed, i.e. written from the –B and –B2.</td>
</tr>
</tbody>
</table>
Reference

- knowledgebase.progress.com/articles/Article/P41910
- knowledgebase.progress.com/articles/Article/17957
Questions?

• dforeman@bravepoint.com
• dforeman@progress.com
• danf@prodb.com / dan@wss.com
• Mobile: +1 541 908 3437
• Please thank the conference organizers for their hard work in putting together an excellent conference