Diagnosing and fixing Firebird performance problems

Alex Koviazin
ak@ib-aid.com

IBSurgeon, www.ib-aid.com
Slowness
Firebird performance problems can be caused by:

1. Bad transactions management
2. Problems in database structure
3. Wrong firebird.conf settings
4. Slow SQL queries
5. Highly concurrent access
Bad transactions management
2 main problems with transactions

1. Long-running [writeable] transactions
2. Rollbacked transactions
Gstat -h

Database header page information:
  Flags                   0
  Checksum                12345
  Generation              1564
  Page size               4096
  ODS version             10.1
  Oldest transaction      10009
  Oldest active           20001
  Oldest snapshot         20001
  Next transaction        25007
  Bumped transaction      1
Long-running transactions

- All transactions have sequential numbers from 1 to…
- Oldest Active Transaction – currently active transaction with minimal number

<table>
<thead>
<tr>
<th>Oldest transaction</th>
<th>10009</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oldest active</td>
<td>20001</td>
</tr>
<tr>
<td>Oldest snapshot</td>
<td>20001</td>
</tr>
<tr>
<td>Next transaction</td>
<td>25007</td>
</tr>
</tbody>
</table>

Interval NEXT – OAT shows number of potentially active transactions – server cannot clean record versions created by these transactions
What is a record version?

Different transactions can see different versions of the record.
Why many record versions are bad

The more versions a record has, the more read operations the server does to find the necessary version.

After data load or restore – no versions.
Why many record versions are bad

• When UPDATE changes indexed fields, indices also must be updated, and - UPDATE does not update keys in the index, it **adds new keys** for new record version!
• DELETE does not delete index keys!

<table>
<thead>
<tr>
<th>N</th>
<th>Tx</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>200</td>
</tr>
</tbody>
</table>

Index key = 100

Index key = 200
How to identify long running active transactions in Firebird?

1. Manual query to MON$ tables
2. HQbird FBDataGuard & Firebird MON$Logger - demo
How to fix active long-running transactions in Firebird?

1. Don’t do it (i.e., fix the source code)
2. Restart connections
   1. stop/start client applications
   2. Restart Firebird
Rollbacked transactions

• When some transaction is marked in transaction inventory as **rollbacked**, it prevents record versions being cleaned by collective or background garbage collection.

<table>
<thead>
<tr>
<th>Oldest transaction</th>
<th>10009</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oldest active</td>
<td>20001</td>
</tr>
<tr>
<td>Oldest snapshot</td>
<td>20001</td>
</tr>
<tr>
<td>Next transaction</td>
<td>25007</td>
</tr>
</tbody>
</table>

**Interval Oldest Snapshot – Oldest Interesting** determines the need for sweep.
How to fix OIT stuck?

1. Don’t do it (always commit 😊)
2. Sweep
   • Sweep is a process of cleaning database from garbage versions
   • Sweep is necessary when OST-OIT interval becomes big

SWEEP reads database from the beginning to the end and cleans obsolete versions
How to make sweep

• Autosweep
  • by default 20000
  • Starts immediately when interval > threshold
  • Slowness at unpredicted moments

• Manual sweep
  • Scheduled sweep during the inactivity period of time
  • Can be run manually with gfix –sweep or in HQbird FBDataGuard
Sweep must be controlled!

• If sweep did not succeed to align transaction markers, it can indicate a serious problem or corruption!

• HQbird FBDataGuard checks sweep status
Problems in database structure
2 main problems with database structure

• Deep indices
• Fragmented tables
Deep indices

- How to find
  - Gstat – r
  - HQbird IBAnalyst – demo

- How to fix – Increase Page size
- In case of 16Kb page size – consider another index
Fragmented tables

- Fragmented by BLOBs
- Fragmented by big records

- How to find – IBAnalyst only
- How to fix
  - Increase page or decrease page size
  - Change schema – move BLOBs to another table
Wrong firebird.conf settings
Wrong firebird.conf settings-1

1. These default settings are too small, must be increased
   • LockHashSlots
   • TempCacheLimit
   • DefaultDBCachePages
   • Situation is better in 3.0
Wrong firebird.conf settings-2

2. Settings which do not correspond Firebird Architecture

- Too big DefaultDBCachePages for Classic/SuperClassic – recommended 256-2048
- Too small DefaultDBCachePages for SuperServer
  - 10000 for 2.5
  - 50k – 2M for 3.0
Wrong firebird.conf settings-3

2. Wrong combination of settings

If FileSystemCacheThreshold < DefaultDBCachePages, file system cache will be disabled
= disaster for Classic/SuperClassic
= not so good for SuperServer
How to fix

1. Read comments in firebird.conf

2. Use optimized Firebird configuration from IBSurgeon
   2. Bundled with HQbird (text files)
Slow SQL queries
Slow SQL queries

How to find slow queries

1. Trace API – 2.5, 3.0
2. MON$ tables – 2.1, 2.5, 3.0
3. FBSscanner – 1.0, 1.5, 2.0, 2.1, 2.5
4. In-app SQL statistics
Trace API

• It catches everything
  • Queries, Transactions, Stored Procedures, Triggers
• It makes all operations slower
  • Can be improved with time threshold, less things to be monitored, etc

• Demo
Show only current SQL queries (no sp/triggers)
  - Idle, Stalled, Active
- It shows reads and writes, not the time
  - Shows also fetches
- Demo
FBScanner

• Works as a proxy between client application and Firebird (3.0 is not supported)
• Can be setup on remote server, and track queries for the selected subset (1 workstation)
How to fix slow SQL query?

• Sorry, it requires 1 day seminar!
Highly Concurrent SQLs
What is highly concurrent SQLs?

- When query which work fine at 1 computer, works 10x-100x slower with many connections
- Lock table -> Mutex wait values is more than 30%
Lock table – where to look

Fb_lock_print –d <database_name>
LOCK_HEADER BLOCK
  Version: 17, Active owner: 0, **Length: 6291456, Used: 5517236**
  Flags: 0x0001
  Enqs: 10906251, Converts: 58907, ReJECTs: 22373, Blocks: 210859
  Deadlock scans: 5841, Deadlocks: 0, Scan interval: 10
  Acquires: 13636997, Acquire blocks: 558879, Spin count: 0
**Mutex wait: 4.1%**
  Hash slots: **2003**, Hash lengths (min/avg/max): **2/ 11/ 26**
  Remove node: 0, Insert queue: 0, Insert prior: 0
**Owners (107):** forward: 26696, backward: 5517140
  Free owners: *empty*
  Free locks (1630): forward: 3878196, backward: 2264580
  Free requests (793): forward: 5412916, backward: 1906516
Lock Ordering: Enabled
Many SQLs compete for the same page
Examples of highly concurrent access

- Implementation of notifications through the table and SELECT
- Update of some flag table
- Getting GEN_ID values very often
How to catch it?

- Use MON$
- Demo
Thank you!
And don’t forget these links

Questions? ak@ib-aid.com

IBSurgewon optimized configuration files

HQbird Standard - tools are compatible with 1.5-3.0, free trial for 14 days
Why should we monitor **lock table parameters**?

- Lock table is a critical part of Classic and SuperClassic architectures
- Access to shared objects is implemented through locks in Lock table...

Analysis of lock table parameters is the easiest way to reveal problems with concurrent access
Why should we monitor lock table parameters?

• Lock table is a critical part of Classic and SuperClassic architectures
• Access to shared objects is implemented through locks in Lock table...

Analysis of lock table parameters is the easiest way to reveal problems with concurrent access
Why should we monitor lock table parameters?

- Lock table is a critical part of Classic and SuperClassic architectures
- Access to shared objects is implemented through locks in Lock table...

Analysis of lock table parameters is the easiest way to reveal problems with concurrent access
How lock table works

Firebird

Page X

How to resolve an access conflict?

Page cache

Page cache

Page cache

fb_inet_server

fb_inet_server

fb_inet_server
How lock table works

All requests to access internal database objects go through lock table.
Why should we monitor lock table parameters?

• Lock table is a critical part of Classic and SuperClassic architectures
• Access to shared objects is implemented through locks in Lock table…

Analysis of lock table parameters is the easiest way to reveal problems with concurrent access
Lock table analysis - raw

- `Fb_lock_print -d <database_name>`
- **LOCK_HEADER BLOCK**
  - Version: 17, Active owner: 0, Length: 6291456, Used: 5517236
  - Flags: 0x0001
  - Enqs: 10906251, Converts: 58907, Rejects: 22373, Blocks: 210859
  - Deadlock scans: 5841, Deadlocks: 0, Scan interval: 10
  - Acquires: 13636997, Acquire blocks: 558879, Spin count: 0
  - Mutex wait: 4.1%
  - Hash slots: 2003, Hash lengths (min/avg/max): 2/11/26
  - Remove node: 0, Insert queue: 0, Insert prior: 0
  - Owners (107): forward: 26696, backward: 5517140
  - Free owners: *empty*
  - Free locks (1630): forward: 3878196, backward: 2264580
  - Free requests (793): forward: 5412916, backward: 1906516
  - Lock Ordering: Enabled
Lock table analysis – where to look

- `Fb_lock_print -d <database_name>`
- `LOCK_HEADER BLOCK`
  - Version: 17, Active owner: 0, **Length: 6291456, Used: 5517236**
  - Flags: 0x0001
  - Enqs: 10906251, Converts: 58907, Rejects: 22373, Blocks: 210859
  - Deadlock scans: 5841, Deadlocks: 0, Scan interval: 10
  - Acquires: 13636997, Acquire blocks: 558879, Spin count: 0
  - **Mutex wait: 4.1%**
  - Remove node: 0, Insert queue: 0, Insert prior: 0
  - **Owners (107):** forward: 26696, backward: 5517140
  - Free owners: *empty*
  - Free locks (1630): forward: 3878196, backward: 2264580
  - Free requests (793): forward: 5412916, backward: 1906516
  - Lock Ordering: Enabled
How to monitor lock table

1) Command prompt (on the server only), run every N min 
   fb_lock_print -d E:\OLTP-EMUL\oltp30.fdb

2) Alerts and automatic recommendations
## Thresholds and recommendations

<table>
<thead>
<tr>
<th>Essential values:</th>
<th>Firebird.conf params to adjust locks:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length: 6291456</td>
<td>LockMemSize = 1048576 x10</td>
</tr>
<tr>
<td>Hash slots: <strong>2003</strong>, Hash lengths (min/avg/max): <strong>2/ 11/ 26</strong></td>
<td>LockHashSlots = 1009 Must be prime number, x20</td>
</tr>
<tr>
<td>Mutex wait: 4.1%</td>
<td>Nothing to adjust, mutex is an indicator of concurrency</td>
</tr>
<tr>
<td>Owners (107)</td>
<td>Number of connections for server</td>
</tr>
</tbody>
</table>

What to monitor?

- Database statistics – Record Versions and Max Versions
- Indices keys
How to monitor

1. With gstat command line tool
2. With HQbird Firebird Database Analyst
Goal of every Firebird developer is to avoid creating versions for records!

Not only because server slow reads multiple record versions, but also because of GARBAGE COLLECTION
What is garbage and how it is collected

• When record versions become obsolete and non-interested by any transaction, it is considered as garbage and need to be cleaned.

• Garbage collection is a process of removing unnecessary records versions
  • It can be cooperative (Classic or SuperClassic) or background (SuperServer)
Why should we monitor garbage collection?

• It consumes resources.
• We should locate and fix parts of the applications which produce many record versions and provoke GC
How to monitor

• 1) Manual SQL queries to MON$ tables
• 2) HQbird Firebird MonLogger
Monitoring transactions in Firebird
Temp files
Temp files

• Temp files are created in default temp folder or in specified folders (TempDirectories in firebird.conf)
• Actually they are written to the disk only if size of all sort files is more than TempCacheLimit parameter
• *It is better to have sorting in memory!*
How to track Temp files

1. Manually check size and quantity of temp files (fb_sortxxx) in all temp folders
2. HQbird FBDataGuard monitors temp files
How to move temp files to memory

• Increase TempCacheLimit parameter
• Warnings:
  • At Classic TempCacheLimit is allocated for each process
  • 32bit processes have 2Gb limitation in memory to address

Use optimized configuration files from IBSurgeon!
Monitor connections, SQL queries and transactions
Connections

• Connection peaks
  - Massive operations
  - Direct web connections
  - Several connection per client

Connections peaks are dangerous at Classic/SuperClassic — memory can be easily exhausted.
Transactions

• Typical mistakes with transactions
  1. Transactions are not closed
  2. Always using 1 transaction per operation (usually due to autocommit)
  3. Using writeable transactions for read-only operations
SQL statements

1. Slow queries
2. Inactive queries (consume memory)

Demo
Hardware monitoring from Firebird point of view
Does your hardware work really good?

• 1. Why we need universal score for hardware

• 2. Firebird Hardware Guide

Universal scoring for Firebird

Official Firebird test!

Available in Firebird repository and in HQbird
Thank you!
And don’t forget these links

Questions? ak@ib-aid.com

IBSurgeon optimized configuration files

Firebird Hardware Guide