DelphiDay italian conference

PIACENZA EDITION

2019 - XVIII Edizione

(almost) 68 ways to optimize Firebird

wintech italia

embarcadero reseller partner

Fabio Codebue

P-SOFT

P-Soft

www.p-soft.biz

www.firebirdsql.it

@fcodebue

linkedin.com/in/fcodebue

@delphiforce

f.codebue@p-soft.biz

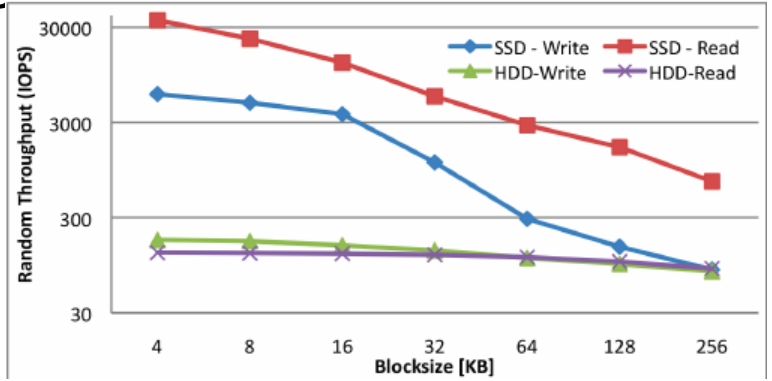
AGENDA

- Hardware optimization
- Firebird server optimization
- Programming optimization
- SQL optimization
- Windows Server optimization
- Linux optimization
- Virtual Server optimization

Hardware Optimization

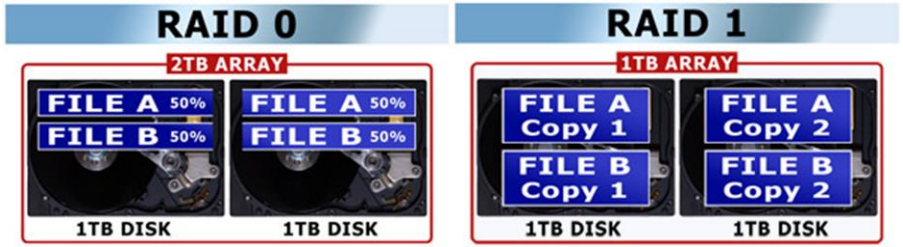
Put database to SSD

- **Put your database on SSD**
- SSD drive provides much better random IO than traditional drives.
- Random IO is critical for reading and writing data distributed through big database file - the majority of database operations require intensive parallel random IO.



RAID

- **Use RAID 10**
  - If you use RAID1 or RAID5, consider RAID10
  - RAID10 it is 15-25% faster.



Under RAID 1, the contents of each file are divided or "stripped" across two or more disks to increase read-write performance. Under RAID 1, a complete copy of each file is stored on two seperate drives, so protecting against a drive failure.

Under RAID 5, data is stripped across two drives, but with parity data (maintaining a record of the differences of the data blocks on each drive) written to a third to allow data recovery in the event of a drive failure. Under RAID 10, data is stripped and mirrored.

Backup Battery Unit

- **Use BBU**
- If you are using RAID controller, check that it has Backup Battery Unit (BBU) installed and operational – some vendors do not provide BBU by default.
- Without BBU, the controller disables the cache, and RAID works very slow, even slower than usual SATA drives.
- Usually, you can check BBU status in the RAID configuration tool.

Write Cache

- Set write cache to **write-back**
- If you are using RAID controller with installed BBU (and server with UPS), check that its cache is set to write-back (not write-through).
- «**Write-back**» enables write cache of the controller.

Read Cache

- Enable read cache
  - If you use RAID controller

check that it has enabled read cache

The page cache is the intermediate between the "working" parts of the database and the disk.

Low level check disk

- Check disk subsystem
  - Check your drives for **bad blocks** and other hardware problems (including **overheating**).
  - Hardware problems can significantly decrease IO performance and lead to database corruptions.

Firebird 2.5

- Use **SuperClassic** or **Classic** in Firebird
  - If you use Firebird 2.5 SuperServer with many connections, try to use SuperClassic or Classic, they can scale better by using all cores of CPU.

Firebird 3.x

- Use **SuperServer** 3.0
  - If you use Classic or SuperClassic in 2.5, consider migration to Firebird 3.0 SuperServer
  - Now it can use multiple cores and combine it with the advantages of the shared cache.

FIREBIRD OPTIMIZATION

Increase page buffers cache

- Increase the size of page buffers cache (parameter *DefaultDBCachePages*) from the default values.
- For 2.5 SuperServer we recommend 10000 pages,
- For 3.0 SuperServer – 50000 pages
- For Classic and SuperClassic – from 256 to 2048 pages.

However, don't set page buffers cache value too high – cache synchronization has its cost, and the idea to put all database into RAM by tuning this value will not work.

Memory and cache

- Increase memory size for sort operations
- Increase the value of *TempCacheLimit* parameter
- It specifies the size of the cache of the temporary space for sorting.
- Default values are too low
  - 8Mb for Classic and 64Mb for SuperServer
- Use at least
  - 64Mb for Classic
  - 1Gb for SuperServer and SuperClassic.

Forced Writes

- Set Forced Writes Off – ATTENTION!!!
- If you have intensive insert or update activity
- You can check it with ***HQbird MonLogger***,
- If you have UPS and replication installed to protect from hardware failures, consider to set Forced Writes settings to OFF, it can increase speed of write operations up to 3 times.

Hash slots

- Increase number of hash slots for Classic/SuperClassic
    - Increase the value of ***LockHashSlots*** parameter for Classic and SuperClassic from the default 1009 to some big prime number (30011, for example)
    - **It will decrease queues in the internal locking mechanism**

CPU Affinity

Firebird 2.5

- Use CPU Affinity for Super Server
  - Set *CPUAffinity* parameter to the value equal to the number of databases in use:
  - SuperServer in 2.5 can use different CPU cores to process requests for the certain databases.
  - CpuAffinityMask = n

The value is taken from a bit map in which each bit represents a CPU.
Thus, to use only the first processor, the value is 1.
To use both CPU 1 and CPU 2, the value is 3.
To use CPU 2 and CPU 3, the value is 6.

Temp space drive

- Use fast drive for temp space
  - Set the first part of ***TempDirectory*** parameter in firebird.conf to the fast disk – SSD or RAM drive.
  - It will decrease the time of big sortings – for example when the database is being restored.

```
TempDirectories = c:\temp;d:\temp
```

Backup location

- Store backups on another drive
  - Store database backups on the dedicated physical drive (RAID).
  - ***It will separate read and write IO during backup***, and increase backup speed and decrease load for the main drive.
  - It is especially important when backups are taken while users are working with the database.

Clean Firebird temporary files

- Temporary files are stored in the following locations:
  - Windows **C:\ProgramData\firebird**,
  - Linux **/tmp/firebird**

- **Normally these files should be cleaned automatically, however**, sometimes it does not happen (for example, in the case of server reboot).
- Check these folders periodically and clean old files – there could be many GBs of outdated files fb_NNN

FileSystemCacheThreshold

- **FileSystemCacheThreshold**.

If FileSystemCacheThreshold is less than DefaultDBCachePages or page buffers, the operating system's file cache will be not used, which can lead to performance problems.
In 99% of cases, it is better to have file cache enabled.
To ensure this, always set parameters according to the following rule: `DefaultDBCachePages = X`

`FileSystemCacheThreshold = X+ N, N>1`

Security DB

- Every connection to the Firebird database establishes the connection to the security database
- **Increase page buffers for securityN.fdb** (empirical optimum is 256 buffers)
- **Move security3.fdb to the fast drive** (it is a standard feature in Firebird 3, in 2.5 it will require reinstallation)
- Then, you can **set Forced Writes OFF** f

FB 3.x superclassic

- **FB 3.x Surperclassic: better performance**

```
To try SuperClassicSet in firebird.conf
     ServerMode=SuperClassic
     DefaultDbCachePages=1024

gfix -buff 0

Restart Firebird

To revert to SuperServerSet in firebird.conf
     ServerMode=SuperServer
     DefaultDbCachePages=N # N*page size*databases_count < 25% RAM
     FileSystemCacheThreshold = N+1

gfix -buff 0

Restart Firebird
```

Big database

- Databases more than 100Gb in 95% cases, it is better to **have the highest available page size**, in order to:
  - **Decrease depth of indices**
  - **Increase utilization of RAM**
  - **Decrease number of system pages**
- To increase the page size, the database should be back up with gbak tool and then restore with parameter –page

```
gbak -c -page 16384
```

no_reserve flag

- The flag **no_reserve** makes Firebird **do not reserve free space (30%) on data pages for the possible record versions**
- This flag allows the data to be stored in a more compact way (and size of the database is less too)
- DB with no_reserve flag operations are slower.
- If your db is not read-only, remove no_reserve flag.

```
Check:      gstat –h database
Disable:    gfix -use reserve database
```

lock table

- **Lock table** is the mechanism of Firebird, which is used to synchronize access to the internal engine objects.
- Can grow automatically, but its increase is a slow
- Lock table can only grow, starting from the initial size
- To prevent multiple rounds of increasing lock table, observe and set in firebird.conf

  LockMemSize=99999999

  LockMemSize on high load systems with ~1000 users is usually below 200Mb.

no_reserve flag

- If

no_reserve flag

- If

no_reserve flag

- If

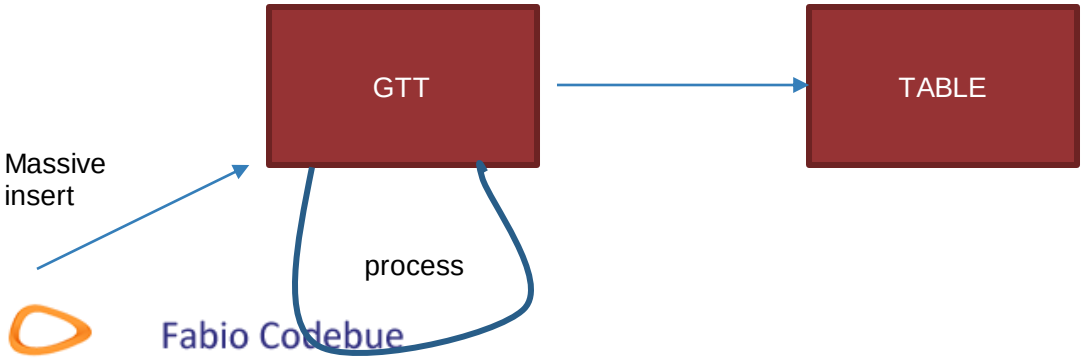
PROGRAMMING OPTIMIZATION

Bulk insert

- Deactivate indices for bulk inserts
  - If you insert or update many records (**more than 25% of the table**), deactivate indices for the table where records are inserted and reactivate them after insert or update.
  - The index rebuild operation can be faster than many updates of the index.

GTT for fast insert

- Use **Global Temporary Tables for fast inserts**
  - To speed up inserts and updates, use Global Temporary Tables for bulk inserts of the large recordsets,
  - and then transfer records into the permanent table.



No more indexes

- **Avoid unnecessary indices**
  - Use fewer indices for tables with intensive inserts and updates.
  - Each index adds significant overhead for insert, update, delete, and garbage collection operations
  - There could be 3-4 additional page reads and writes when the single record is being inserted, updated, deleted, cleaned for each index.

Don’t use UDF

- Replace UDFs with embedded functions calls
- Many embedded functions were added in the recent versions of Firebird, which offer functionality previously available only in UDF libraries.
- Replace such functions where possible, since embedded functions work up to 3 times faster than UDFs.
- **Deprecated in Firebird 4.x**

Read Transactions

- Use **read-only transactions for read operations**
  - Use read-only transactions for operations which do not change record (i.e., SELECTs) with isolation mode = read committed.
  - Such transactions do not retain record versions from the garbage collection, and can run indefinitely
  - **They do not affect database performance.**

Write Transactions

- Use short write transactions and get rid of ALL long-running (for operations INSERT,UPDATE, DELETE).
- **The shorter writeable transaction is, the better.**

The short transactions retain proportionally less number of record versions from garbage collection than long-running. Unfortunately, even the single long-running transaction (from the development tool left open, for example) can screw the good effect of all other short writeable transaction.

That's why you need to monitor long-running transaction and fix the appropriate places in the source code.

Long record chains

- Avoid situations when one record has many record versions
- Firebird works much slower with long record chains. (to see how many record versions some tables has, and what is the longest record chain you can use **HQbird IBAnalyst** tool, tab Tables, sort on "Max Version").
- Use the combination of inserts and scheduled delete of old records instead of multiple updates of the same record.

SQL OPTIMIZATION

Use Prepare

- **Use PREPARE correctly**
  - Use prepared statements to run SQL queries where only parameters are changed

for example, make prepare before the loop of such queries.

  - Prepare can take significant time (especially for big tables), and **preparing the query only once will greatly increase the overall performance**.

Don’t abuse Prepare

- Do not keep queries in the prepared state without the necessity
- The vast majority of them runs only once, and then they just sit in the RAM, making Firebird working set bigger, and slow down MON$ queries.
- The recommendation **is to keep SQL queries in the prepared state only they are intended to be started many times, or if their prepare time is big**

Always close queries with large sorting

- *Until SQL query with sorting (ORDER BY, GROUP BY, UNION, distinct) is not closed, Firebird retains sorted records in the memory.*
- TempCacheLimit allocated the size of memory in firebird.conf, by default it is 64Mb.
- The recommendation is to close all such queries in a timely manner.

Commit when needed

- Don't COMMIT too often during bulk insert/update operation
  - In the case of bulk INSERT/UPDATE/DELETE operation, **don't commit the transaction after each change**
  - It can happen if you are using auto commit option in your database driver
  - Commit transactions at least after 1000 operations or more.
  - Each transaction commit runs several read/write IO operations against the database, that's why **often commits decrease database performance.**

Long record chains

- **"Turn off" indices if you are using IN** with many constants
  - If you are using construction

```
WHERE fieldX IN (Constant1, Constant2,… ConstantN)
```

and there is an index on fieldX, Firebird will use an index as many times as many constants are in the IN list.
Disable index search by turning fieldX into expression +0:

```
WHERE fieldX+0 IN (Constant1, Constant2,… ConstantN)
Or  for strings  use fieldX||''
```

Use JOIN instead IN

- **Replace IN with JOIN**
  - Avoid using queries with nested WHERE
    `IN(SELECT... WHERE IN (SELECT.. WHERE IN() ))`
  - It can confuse Firebird optimizer.
  - **Transform nested INs into joins.**

LEFT or not LEFT JOIN

- Use LEFT JOIN in the correct way
    - If you are using LEFT OUTER joins

    explicitly put tables in the join **from the smallest one to the largest one**.

Avoid unnecessary LEFT JOINs

Sample:

```
T1 LEFT JOIN T2 ON (…) WHERE T2.Field_condition
```

- Change LEFT JOIN to INNER JOIN
- INNER JOIN gives more freedom for Firebird optimizer, and it is optimized much better than LEFT.
- Especially it makes sense in the following cases
  - No condition for T1 in WHERE clause
  - T2 is a small table

FIRST, SKIP and ROWS

- **Limit fetch of SELECT queries**
  - Always try to limit the large output for SELECT queries

with FIRST… SKIP or ROWS clauses.

  - If the query is not designed specifically as a report (which requires all records to be printed/exported), usually it is enough to show top 10-100 records.
  - **Fetch only necessary records.**

SQL projection

- Specify less number of columns in SELECT with ORDER BY/GROUP BY
- Reduce the number of columns and their summary width in queries with ORDER BY/GROUP BY both in SELECT part (i.e., fields to be shown) and in the ORDER BY
- Firebird merges columns from SELECT and ORDER BY/GROUP BY clauses and sorts them in memory (or, if memory is not enough, on the disk).
- A long VARCHAR in SELECT, the size of the sort files can be really large (many gigabytes).

Derived tables

- Use **derived tables to optimize SELECT** with ORDER BY/GROUP BY
  - Another way to optimize SQL query with sorting is to use derived tables to avoid unnecessary sort operations.

```
SELECT FIELD_KEY, FIELD1, FIELD2, ... FIELD_N
FROM T
ORDER BY FIELD2
```

```
SELECT T.FIELD_KEY, T.FIELD1, T.FIELD2, ... T.FIELD_N
FROM (SELECT FIELD_KEY FROM T ORDER BY FIELD2) T2
JOIN T ON T.FIELD_KEY = T2.FIELD_KEY
```

VARCHAR or BLOB

- Store short strings in VARCHAR, large in BLOBs
  - To store short character data, use VARCHARs,
  - To store long texts, use BLOBs.

Varchars are faster for the small pieces of data because they are stored in the record, and the whole record is read during the same IO cycle, and if record size is less than 2/3 of the database page size, the whole record is stored on the same database page.

BLOBs are stored outside of the record, and require the additional round of IO to read it, and they show the advantage with reading and writing long strings.

BLOB

- **Exclude BLOB columns from the large SELECTs**

- Use a kind of late binding with sub-selects to selectively show information from BLOBs

(for example, show the content of the document).

Primary Key - BIGINT

- Use BIGINT type for
  - auto-incremented primary
  - unique keys
  - identifiers of all types.
- **Operations with BIGINT are the fastest**
- BIGINT has enough capacity to store almost all data ranges.

Primary key - VARCHAR

- **Don't use VARCHARs for keys**
  - Don't use VARCHAR for identifiers unless it is really necessary
  - Operations with them are far less effective than with integer columns.
  - **Especially avoid GUIDs, as identifier** due to the random distribution of GUID values INSERT/UPDATE operations with Primary/Unique Keys GUIDs can be 20 times slower than with integers.

Indexes stats

- **Recalculate indixes statistics regularly**
- Update indices statistics for the tables with frequent or massive changes with command **SET STATISTICS**
- It allows Firebird optimizer to choose better SQL plans.

- **HQbird Firebird DataGuard** can perform such recalculation of indices statistics automatically according to the desired schedule (usually once a week).

Connection pool

- **Use connection pool**
- If database connections to Firebird database are short use connection pool

typical for web-app in PHP use

```
function ibase_pconnect
```

instead of

```
ibase_connect
```

LINGER (FB 3-x)

Firebird 3.0

- **Use LINGER option**
- Database connections are short
- LINGER option to **keep cache active during the specified amount of time**
- It will keep frequently used pages in the cache even if there will be no other connections.

For example, *ALTER DATABASE SET LINGER TO 60* will keep the cache for 60 seconds after the end of the last connection.

HASH JOINs (FB 3.x)

Firebird 3.0

- **Use HASH JOINs**
- Case the of joining big and small tables, **HASH JOIN could be much faster than normal join** which uses «nested loop» with index.
- To make Firebird optimizer to use HASH join, use +0 in the join condition:

```
T1 JOIN T2 ON T1.FIELD1+0 = T2.FIELD2+0
```

PSQL

Firebird 3.0

- Mark your PSQL functions which do not have parameters and return constant values with keyword **DETERMINISTIC**.

- The deterministic functions are calculated and cached in the scope of the current query.

Analytical functions

Firebird 3.0

- **Use analytical (window) functions**
- If you are running SELECT with simultaneous output of some column and aggregated function for it, use window (analytical) functions
- it is faster than the subquery or 2 queries

```
SELECT id, department, salary, salary /
(select sum(salary) from employee) percentage
FROM employee
```

```
SELECT id, department, salary, salary /
sum(salary) OVER () percentage
FROM employee
```

gbak

- Use **switch -se** for gbak
  - Use switch *–se* to increase gbak backup and/or restore speed up to 20%

```
gbak -b -g -se service_mgr c:\db\data.fdb
    e:\backup\data.fbk
```

Long record chains

- **WHERE CURRENT OF**
- The fastest way to process records fetched by the cursor in PSQL is the clause

```
where current of <>
```

It is faster than

```
where rb$db_key = :v_db_key
```

and much faster than search with a primary or unique key.

MON$

- **Avoid often queries to monitoring tables**
  - Don't run queries to Firebird monitoring tables (MON$) too often

such queries consume significant resources and can greatly decrease the performance of the main business logic.

Recommend running MON$ queries not often than once per minute.

NO_AUTO_UNDO

- Use **NO_AUTO_UNDO option for bulk inserts/updates**
  - If you are running many DML (Update/Insert/Delete) commands in the frames of the same transaction, Firebird merges undo-log of each command with undo-log of the transaction.
  - To speed up bulk DML operations start the transaction with «NO AUTO UNDO» option, in order to **do not merge undo-logs of each command with the transaction's undo-log.**

SRP Auth

Firebird 3.x

- Do not use SRP authentication in if you don't need it

    - Does not use SRP users authentication if you don't really need it
    - Connect with SRP authentication is established slower than the regular connection.

Avoid unnecessary counting of records

- Common mistake is to use select count() just to check the existence of the record.

```
(select count(*)…. where condition1) >0
```

Better use this construction instead

```
Exists(select first 1 id where condition1)
```

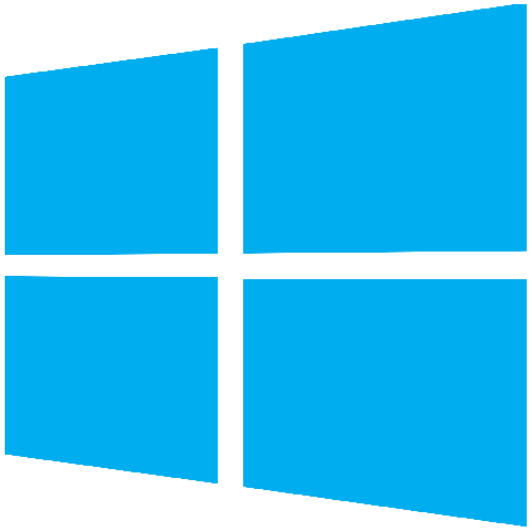
If condition1 returns more than 1 record, the proposed option will be much faster, because it does not read all records, it stops after the first fetched record.

Avoid unnecessary sorting in SP

- The ordering of the query's results inside the stored procedure should be justified by the business logic.

- Check your PSQL code for similar situations and remove useless ORDER BY (as well as distinct and UNION).

```
create or alter procedure
NEW_PROCEDURE
returns (
    SUMX double precision)
as
declare variable _amount
double precision;
begin
for select T1.amount from
Table1 t1 where ....
    order by id
    into :_amount
    do
    begin
     sumx=sumx+_amount
    end;
  suspend;
end
```



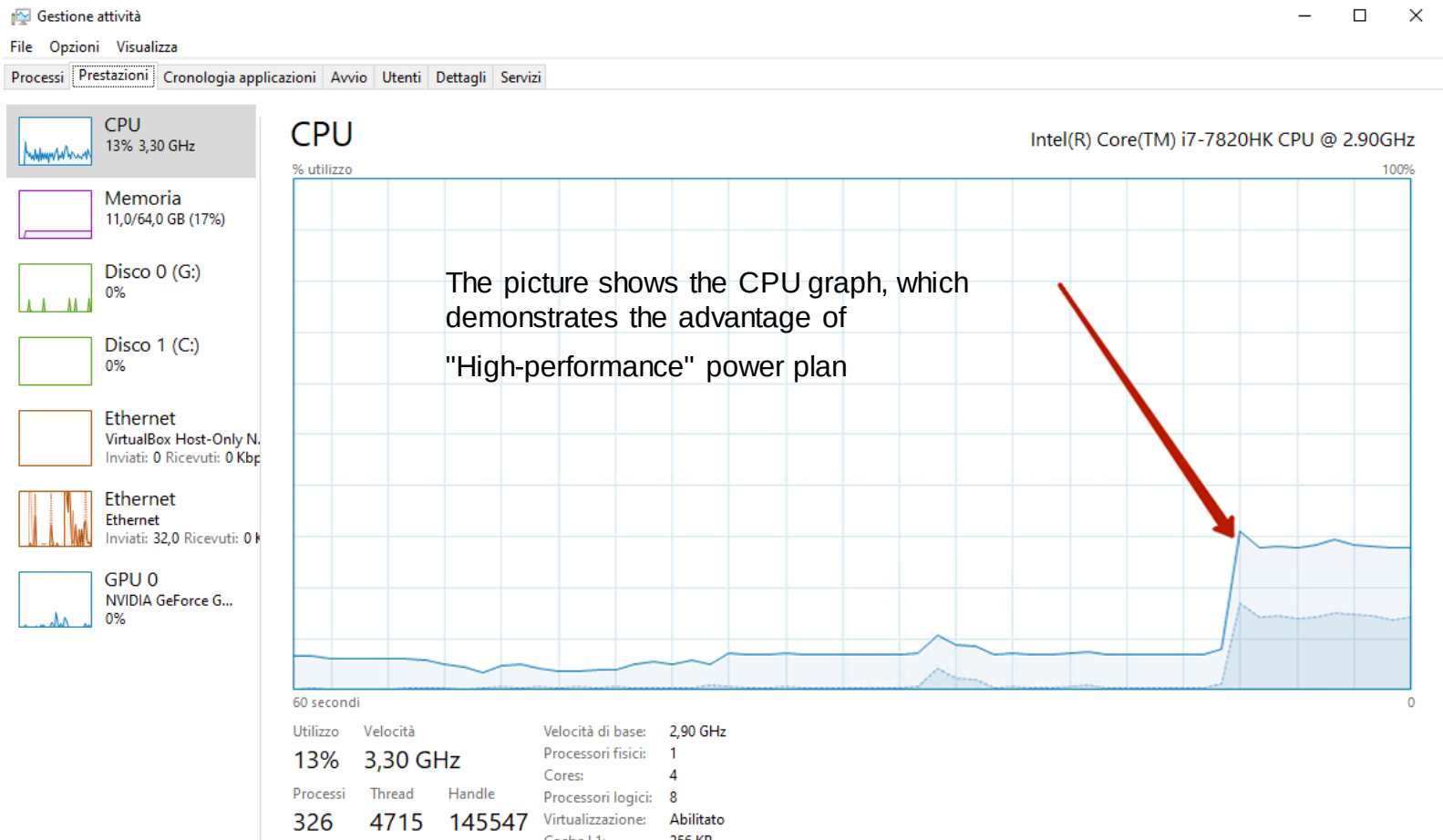
Windows Server

OPTIMIZATION

Windows power plan

- Windows Server power plan default set to **Balanced**
  - It is not suitable for database servers.
- Set it to **High Performance** **and gain approximately +20% of the performance for CPU-intensive operations**.
- It can be set online, without restart or reboot

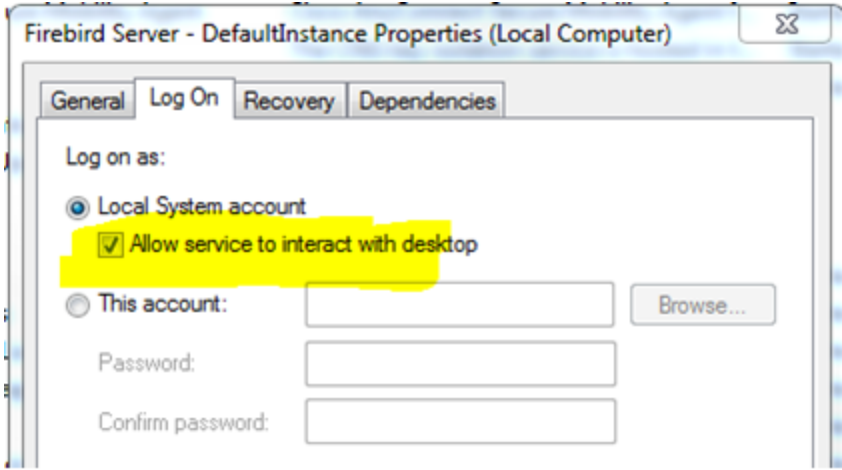
Windows power plan



delphiForce

Interact with desktop

- **Firebird with Classic architecture on Windows**
  Enable «Interact with desktop»
- *Without this setting, the resource «desktop heap» is limited by Windows*
- Firebird cannot open more than 250-300 connections (depends on the metadata of the database and related memory consumption) – there will Out Of Memory error.



Beware of Domain Controller

- Windows with Domain Controller role **disables the write cache on the disk** with Active Directory database.
- We have significantly worse performance than on the servers without Active Directory roles.
- Please note, that this problem affects such popular Windows version as Windows Small Business Server 2011, as well as other versions with DC.

RAM for file cache

- OS Memory Manager has implications regarding the memory allocation, and, by default, Windows requires 40% of RAM for file cache.
- Unfortunately, the poor tool Windows Task Manager shows memory, which is used for file cache as «free», and some administrators try to make Firebird consume all this free memory, so they set DefaultDBCachePage parameter in firebird.conf to very high values, and it usually leads to swapping.

RAM for file cache

- Always use RAMMap tool to see the actual memory usage on Windows.
- **Firebird server RULE**

$$FBmemws \leq RAM$$

Firebird memory (Working Set) should be less than 40% of total RAM.

If the total size of working sets for all processes is more than 50%, swap can be started by Windows.

LINUX OPTIMIZATION

Max open files

Increase «max open files» limit on Linux

- Check limits for your Firebird process (SuperServer or SuperClassic) with the following command:

```
cat /proc//limits
```

pay attention to the line with the max number of open files.

- Firebird can use up to 4 handles per connection, and if you something like this:

```
Max open files 4096 4096 files
```

it means that the total number of connections served by the Firebird process will be limited to something around 1000.

User modern Linux

- Good performance improvement after the migration from
  - CentOS 6 to 7,
  - Ubuntu 12 to 16 (on the same hardware!),
- now **it is the must-have recommendation for database servers with more than 250-300 connections.**
- Recommended versions of Linux:

  CentOS 7.x and Ubuntu 16, 18.

RAM for file cache

- Linux works with file cache in another way than Windows,
- The amount of RAM used for file cache can be significantly less, than on Windows, without noticeable performance Firebird degradation.
- However, **to guarantee high performance of the system with the high number of connections**, especially on Classic and SuperClassic, the good idea will be to **reserve 30% of RAM for file cache**.

irqbalance

Prerequisite: servers with the high number of cores.

- **irqbalance** often improves
  - Firebird performance
  - CPU load balancing

virtual server

Memory Overcommit

- VM can be configured to have more memory than physically exists on the host machine – with a feature known **Memory Overcommit**
- It means that in the case of a peak of memory consumption (on VM with the database server or on the neighbor VM) swap can start, which will lead to significant delays.
- For high-performance VM intended for a database server, all memory should be static.

Check VM limits

- Often VMs are created with default CPU and IO limits which can be very low, like 50 IOPS and 10% CPU.

- Check your server VM settings and remove any limits - high-performance database server should have all possible CPU, bandwidth, and IO.

GRAZIE…

Sostienici, diventa un associato …
https://firebirdsql.org/en/membership/

Fabio Codebue

P-Soft

www.p-soft.biz

www.firebirdsql.it

@fcodebue

linkedin.com/in/fcodebue

@delphiforce

f.codebue@p-soft.biz

Toolz