Firebird and IBExpert White Paper

ibec_StartTraceSession

Fikret Hasovic, August 2022

As we have previously explained, IBEBlock is a set of DDL, DML and other statements that are executed on the server and on the client side, and which include some specific constructions applicable only in IBExpert or IBEScript (excluding the free versions of these products), independent of the database server version.

IBEBlock is an extremely complex and capable piece of software, which can help you to increase your productivity and save both time and money spent on your projects.

Now, imagine you have a database under heavy load, and you would like to identify which attachment is responsible for that. You can execute a select on the *MON$* tables, which will show most active connections and if one of the connections has an unusually high load compared to all others, this connection should be traced into a log.

Trace and audit

If you are already a Firebird user, you have probably heard about Firebird's *Trace and Audit* services, initially developed from the TraceAPI contributed by Nickolay Samofatov, which he had developed for Red Soft Database. This feature enables you to look inside the Firebird engine and log various events for real-time analysis.

Two different kinds of traces can be performed: user traces and a system audit.

There are three general cases for use:

- Constant audit of engine activity
- Interactive trace of specified activity in specified database(s).
- Engine activity logging for a specified period of time (be it hours or a whole day) for later analysis.

For more detailed explanation, please see
https://www.ibexpert.net/ibe/pmwiki.php?n=Doc.TraceAndAudit

Nevertheless, an audit/trace does not replace the monitoring tables, because with the monitoring tables, the user can query a snapshot of the current state of a database, whereas an audit/trace can be compared with a sequential stream of logged statements. For example, if you wish to know the number of currently active transactions inside a database, simply query the *MON$TRANSACTIONS* table. With the audit/trace facility, this probably can only be done with advanced data analysis tasks on the log data stream (if at all).

The IBExpert release version 2022.06.10 introduces *ibec_StartTraceSession* (among other useful new features and improvements).

ibec_StartTraceSession starts a trace session using the Firebird Services API and writes trace data into a file. Additional filtering of trace items is possible via a callback block.

Syntax

```
function ibec_StartTraceSession(ConnectParams : string; TraceConfig : string;
OutputFile : string; FilterBlock : variant; ProgressBlock : variant) : variant;
```

ConnectParams - a list of connection params and some additional options delimited with a semicolon (;):

- *Server* - server name including port number if necessary.
- *User* - user name.
- *Password* – password.
- *ClientLib* - path to a client library dll.
- *MaxFileSize* - maximum size of the result file, in megabytes. If not specified the result file size will be limited to 100 megabytes.
- *StopAfter* - determines the duration of the trace session in minutes. The default value is 60 minutes.
- *IncludeStatistics* - some statistical data (total and max per second number of processed lines and events) will be written at the beginning of the trace data.
- *AppendToExisting* | *AppendMode* - if specified, trace data will be added to the existing file (if it exists).

TraceConfig - Firebird trace configuration.

OutputFile – the name of the result file which will contain the trace data.

FilterBlock - a callback block which performs additional filtering of trace items if necessary. This block is called for every trace item and accepts a trace item text as an input parameter (*EventSource*).
The output parameter *NeedSkip* determines whether the trace item should be skipped, i.e. not included into the result file.

If no additional filtering is needed, pass NULL or an empty string as the *FilterBlock* value.

ProgressBlock - a callback block that is intended to display the progress of tracing. It receives an array of the following statistical data:

- *Data[0]* (or *Data['LinesProcessed']* - number of processed lines.
- *Data[1]* (or *Data['EventsProcessed']*) - number of processed trace items/events.
- *Data[2]* (or *Data['EventsMatched']*) - number of trace items written to the result file.
- *Data[3]* (or *Data['OutputSize']*) - current size of the result file in bytes.

The output parameter *Threshold* determines a delay in milliseconds before the next call of *ProgressBlock*.

ibec_StartTraceSession returns NULL if a trace session is started successfully, otherwise it returns an error message as a result.

One example of usage can be the following:

```
execute ibeblock
as

  -- Filter block
  declare ibeblock filter_block (EventSource variant)
  returns (
    NeedSkip boolean = FALSE)
  as
  begin
    -- Using regular expression
    NeedSkip = ibec_preg_Match('(?i)0\x20ms', :EventSource);

    -- Using ibec_Pos
    NeedSkip = ibec_Pos(' 0 ms', :EventSource) > 0;
  end;

  -- Callback block
  declare ibeblock progress_block (data variant)
  returns (
    threshold integer = 2000)
  as
  begin
    for i = 0 to ibec_High(data) do
      data[i] = ibec_Cast(data[i], __typeString);

    s = ibec_Format('| %15.15s | %16.16s | %14.14s | %16.16s |', data[0],
data['EventsProcessed'], data['EventsMatched'], data['OutputSize']);
    ibec_ProgressEx(s, __poNoCRLF + __poReplaceLast);
  end;

begin
  sConfig = ibec_LoadFromFile('D:\temp\trace.conf');
  sOutFile = 'D:\Temp\tracedata.txt';
  sConnectParams = 'Server="myserver/3044"; User=SYSDBA; Password=masterkey;
ClientLib="D:\FB4_CLIENT\fbclient.dll"';
  sOptions = 'MaxFileSize=10; StopAfter=60; IncludeStatistics';

  ibec_Progress('');
  ibec_Progress('+-----------------+------------------+----------------+--------
----------+');
  ibec_Progress('| Lines processed | Events processed | Events matched | Output
file size |');
  ibec_Progress('+-----------------+------------------+----------------+--------
----------+');

  Res = ibec_StartTraceSession(:sConnectParams || ';' || :sOptions,
                               :sConfig,
                               :sOutFile,
                               filter_block,
                               progress_block);

  if (Res is not null) then  -- Error caused
```

```
    ibec_Progress('ERROR: ' || Res);
end;
```

Let's now return to our original task, to identify which connection generates an unusually high database load.

If you are an experienced user, you can try a select on any combination of *MON$* tables to get interesting attachment IDs as results, and you can later change your query.

My approach here will be to store the current state of page reads/write in a new table, for example every minute and check which attachments performed the largest amount of reads/writes within the last 10 minutes and is not already actively traced.

Also, if the total number of *reads* + *writes* is below 100,000 within last 10 minutes, it should be ignored.

I have created a table in the database to hold the data collected every minute:

```
CREATE TABLE IBE$MON (
    ATTACHMENTS_ID  BIGINT NOT NULL,
    STATEMENT_ID    BIGINT NOT NULL,
    TRANSACT_ID     BIGINT,
    OPERATIONS      BIGINT,
    TS              TIMESTAMP DEFAULT current_timestamp
);
```

The following indices will help with our query later:

```
CREATE INDEX IBE$MON_IDX1 ON IBE$MON (ATTACHMENTS_ID);
CREATE INDEX IBE$MON_IDX2 ON IBE$MON (TS);
```

Now I have created *ibeCollectData.sql* as follows:

```
execute ibeblock
  as
  begin
    FBSrc  =
ibec_CreateConnection(__ctFirebird,'DBName="localhost:C:\db\mydb.fdb";
                                  ClientLib=c:\db\firebird3\fbclient.dll;
                                  user=SYSDBA; password=masterkey; names=UTF8;
sqldialect=3');
    ibec_UseConnection(FBSrc);

      insert into ibe$mon (
        attachments_id,
        statement_id,
        transact_id,
        operations)
      select st.mon$attachment_id,
        st.mon$statement_id,
        st.mon$transaction_id,
        cast(sum(io.mon$page_reads)+
        sum(io.mon$page_writes) as integer)
      from mon$statements st
      join mon$attachments a on a.mon$attachment_id = st.mon$attachment_id
      join mon$io_stats io on (st.mon$stat_id = io.mon$stat_id)
      group by 1, 2, 3
      having (sum(io.mon$page_reads) + sum(io.mon$page_writes)) > 0;

      COMMIT;

    ibec_CloseConnection(FBSrc);
  end
```

To execute the previous script, I have created *ibeCollectData.cmd*, which is basically a batch script:

```
ibescript.exe ibeCollectData.sql
timeout 60
ibeCollectData.cmd
```

This batch script will execute the IBEBlock script every minute (adjustable, see timeout value) and repeat, so that you will have a snapshot of the *MON$* tables every minute and stored in the table for time-related purposes.

```
execute ibeblock
as
begin
    MyFunc = 'EXECUTE IBEBLOCK
              AS
              BEGIN
                sCRLF = ibec_CRLF();
                sConnectParams = ''Server="127.0.0.1/3050"; User=SYSDBA;
Password=masterkey; ClientLib="c:\db\firebird3\fbclient.dll"'';
                sOptions = ''MaxFileSize=10; StopAfter=10; IncludeStatistics;
AppendToExisting'';
                sConfig = ''database '' || sCRLF ||
                            ''{'' || sCRLF ||
                            ''enabled = true '' || sCRLF ||
                            ''connection_id = '' || att_id || sCRLF ||
                            ''log_statement_prepare = true '' || sCRLF ||
                            ''log_statement_free = true '' || sCRLF ||
                            ''log_statement_start = true '' || sCRLF ||
                            ''log_statement_finish = true '' || sCRLF ||
                            ''time_threshold = 0 '' || sCRLF ||
                            ''}'';
                sOutFile = ''c:\Temp\tracedata_'' || att_id || ''.txt'';

                Res = ibec_StartTraceSession(:sConnectParams || '';'' ||
:sOptions,
                                :sConfig,
                                :sOutFile,
                                null,
                                null);

                if (Res is not null) then  -- Error caused
                  ibec_Progress(''ERROR: '' || Res);
              END';

    FBSrc  = ibec_CreateConnection(__ctFirebird,'DBName="localhost:C:\db\
mydb.fdb";
```

```
                 ClientLib=c:\db\firebird3\fbclient.dll;
                 user=SYSDBA; password=masterkey; names=UTF8; sqldialect=3');
    ibec_UseConnection(FBSrc);

         for select IBE$MON.attachments_id, sum(IBE$MON.operations) io from
IBE$MON
         where (IBE$MON.ts > dateadd(-10 minute to current_timestamp))
         group by IBE$MON.attachments_id
         into :att_id, :IO do begin
           if (IO > 10000/*or any other threshold load value*/) then begin

             TFunc = replace(MyFunc, 'att_id', cast(att_id as varchar(20)));
             fn = cast(att_id as varchar(20)) || '.ibe';
             ibec_SaveToFile(fn, TFunc, __stfOverwrite);
           end
         end
         COMMIT;


     ibec_CloseConnection(FBSrc);
  end
```

As with the previous script, I have created an *ibeTraceLoad.cmd* script:

```
ibescript.exe ibeTraceLoad.sql
for %%i in (*.ibe) do start "" "ibescript.exe" "%%~i"
timeout 10
del *.ibe
timeout 50
ibeTraceLoad.cmd
```

If you compare this IBEBlock script to the one at the beginning of this document, you will notice that I have constructed trace options in code, instead of using the trace config file:

```
sConfig = ibec_LoadFromFile('D:\temp\trace.conf');
```

in comparison to:

```
sConfig = ''database '' || sCRLF ||
                           ''{'' || sCRLF ||
                           ''enabled = true '' || sCRLF ||
                           ''connection_id = '' || att_id || sCRLF ||
                           ''log_statement_prepare = true '' || sCRLF ||
                           ''log_statement_free = true '' || sCRLF ||
                           ''log_statement_start = true '' || sCRLF ||
                           ''log_statement_finish = true '' || sCRLF ||
                           ''time_threshold = 0 '' || sCRLF ||
                           ''}'';
```

However, you can use your preferred method of specifying the trace config.

In a nutshell, our main script here, **ibeTraceLoad.sql**, will check the history of the *MON$* tables from our table for the last 10 minutes, and if the threshold value is exceeded, it will generate individual scripts so that trace sessions can be started in parallel if there is more than one connection generating an abnormal load on the database.

These individual scripts are, in this case, named after the attachment ID, with a file extension *ibe*.

Since trace sessions will be "suspended" until the end of the trace session, I couldn't execute `ibec_StartTraceSession` in a loop, so I used a loop to create separate IBEBlock scripts, and used the command

```
for %%i in (*.ibe) do start "" "ibescript.exe" "%%~i"
```

to execute all generated scripts that are found. The batch script will wait 10 seconds after running the scripts and then try to delete old *.ibe* files, so that out-of-date scripts are removed. Please note that `ibec_StartTraceSession` is run for 10 minutes, but that value can be adjusted for your own needs.

I decided to check for high load connections once every minute, but if a trace session is in progress, it will skip it. After an additional sleep period of 50 seconds, the script will repeat itself. If there is already a trace session in progress, a new one cannot be started before the trace session (which is specified in the call of the function) has expired for 10 minutes.

Also, I decided to use multiple files to output trace results, for each attachment generating abnormal load there is separate trace log, but you can use single file if you prefer.