Measuring Firebird Disc I/O

Paul Reeves

IBPhoenix

IBPhoenix
THE POWER WITHIN

Introduction

Disc I/O is one of the main bottlenecks in Firebird. A good disc array can give a massive increase in available IOPS. The question is how do we measure our disc I/O requirements?

Two Basic Ways To Measure IO

- From within Firebird itself
- From the host O/S

Measuring I/O from Firebird

- Firebird knows what it has done – MON$IO_STATS allows us to catch this info.
- Portable across platforms and architecures
- Works wih all versions since Fb 2.1
- Sort of a work in progress -enhanced in 2.5

MON$IO_STATS

Firebird tracks four IO counters

- Page Reads - physical reads from disc
- Page Writes – physical writes to disc
- Page Fetches – pages read from memory
- Page Marks – pages marked in memory for writing to disc.

About Fetches

Mostly we can ignore Fetches but be sure to make sure that the cache is big enough otherwise excessive reads will occur.

Be sure to tune DB Cache before drawing any conclusions about Firebird disc I/O.

Understanding How the MON$ tables work

- All the Firebird monitoring tables run within snapshot table stability transactions.
- You need to start a new transaction to see the latest data in the monitoring tables.
- Historic data is not stored.
- Once a transaction is finished or an attachment is disconnected the details disappear from the monitoring tables.
- Cumulative database level stats requires persistent connections. This means that even database level stats are lost if all connections to the database are closed.

Making Stats persist - DBTriggers

- DB Triggers available at Connection and Transaction Level.
- Stats are stored at Attachment and Transaction Level (amongst others.)
- Solution – catch stats at end of Connection or Transaction.

Capturing Txn Stats

- Initial analysis indicated that txn level stats may not be complete – more study required.
- Worse – Txn commit triggers and read only transactions don't mix.

Capturing Connection Level Stats

- Stats seem to be complete(ish).
- No problems with r/o txns.

A table to store the data

```
CREATE TABLE PERSISTENT_IO_STATS (
PERSISTENT_IO_STATS_ID INTEGER NOT NULL,
LOG_TIME TIMESTAMP
DEFAULT CURRENT_TIMESTAMP NOT NULL
MON$STAT_ID INTEGER,
MON$STAT_GROUP SMALLINT,
MON$PAGE_READS BIGINT,
MON$PAGE_WRITES BIGINT,
MON$PAGE_FETCHES BIGINT,
MON$PAGE_MARKS BIGINT,
MON$REMOTE_ADDRESS VARCHAR(253),
MON$REMOTE_PROCESS VARCHAR(253),
MON$USER CHAR(31),
MON$ATTACHMENT_ID INTEGER,
DURATION BIGINT,
CONSTRAINT PK_PERSISTENT_IO_STATS PRIMARY
KEY (PERSISTENT_IO_STATS_ID)
);
```

And a trigger to capture it with...

```
END_TIME = cast('NOW' as timestamp);

SELECT MIO.MON$STAT_ID, MIO.MON$STAT_GROUP, MA.MON$TIMESTAMP
      FROM MON$ATTACHMENTS MA
      JOIN MON$IO_STATS MIO on MIO.MON$STAT_ID = MA.MON$STAT_ID
      WHERE MON$ATTACHMENT_ID = CURRENT_CONNECTION
into :stat_id, stat_group, :start_time;

DURATION = datediff ( millisecond from :START_TIME TO :END_TIME);

if (stat_group = 1) then begin
      insert into PERSISTENT_IO_STATS (MON$STAT_ID, MON$STAT_GROUP, MON$PAGE_READS,
             MON$PAGE_WRITES, MON$PAGE_FETCHES, MON$PAGE_MARKS, EVENT_DESCRIPTION,
             MON$REMOTE_ADDRESS, MON$REMOTE_PROCESS, MON$USER, MON$ATTACHMENT_ID,
             DURATION)

      select MIO.MON$STAT_ID, MIO.MON$STAT_GROUP, MIO.MON$PAGE_READS,
             MIO.MON$PAGE_WRITES, MIO.MON$PAGE_FETCHES, MIO.MON$PAGE_MARKS, 'DISCONNECT',
             MA.MON$REMOTE_ADDRESS, MA.MON$REMOTE_PROCESS, MA.MON$USER,
             MA.MON$ATTACHMENT_ID, :DURATION
      from MON$IO_STATS MIO
      join MON$ATTACHMENTS MA on MIO.MON$STAT_ID = MA.MON$STAT_ID
      where MIO.MON$STAT_ID = :stat_id;
end
```

Add a couple of Stored Procs...

- And we can start to analyse our disc I/O by Application, User, IP address.

Add a couple of Stored Procs...

And we can start to analyse our disc I/O by Application, User, IP address.

ID	RIOPS	WIOPS	DURATION	APPNAME	USERNAME	IP_ADDRESS
82	0	0	503661	Files\Firebird\Firebird_2_1\bin\isql.exe	DBOWNER	192.168.0.13
113	0	7	3	httpd2-prefork	DBOWNER	127.0.0.1
114	0	7	3	httpd2-prefork	DBOWNER	127.0.0.1
101	0	0	260623	/firebird/bin-superserver/isql	DBOWNER	192.168.0.2
76	0	8	3	httpd2-prefork	DBOWNER	127.0.0.1
77	0	38	2	httpd2-prefork	DBOWNER	127.0.0.1
78	0	11	2	httpd2-prefork	DBOWNER	127.0.0.1
79	609	5	3	/firebird/bin-superserver/gbak	SYSDBA	127.0.0.1
27	0	0	102602	Files\Firebird\Firebird_2_1\bin\isql.exe	DBOWNER	192.168.0.13
83	166	13	1	lamerobin	DBOWNER	127.0.0.1
84	0	0	16	lamerobin	DBOWNER	127.0.0.1
92	81	7	3	httpd2-prefork	DBOWNER	127.0.0.1
93	41	41	2	httpd2-prefork	DBOWNER	127.0.0.1
94	3	11	2	httpd2-prefork	DBOWNER	127.0.0.1
97	2094	2	4	/firebird/bin-superserver/gbak	SYSDBA	127.0.0.1
85	0	0	90320	lamerobin	DBOWNER	127.0.0.1
99	2838	2	3	/firebird/bin-superserver/gbak	SYSDBA	127.0.0.1
102	2809	2	3	/firebird/bin-superserver/gbak	SYSDBA	127.0.0.1
104	56	4	5	httpd2-prefork	DBOWNER	127.0.0.1

Problems with MON$IO_STATS

- Attachment level stats are almost complete – except that SS doesn't include back-ground threads.
- Accurate stats can only be acquired by running Classic.
- Long running attachments will produce rub-bish stats if tracking IOPS is the goal.
- But well written applications do not leave connections open – do they?.

Using the host O/S for monitoring

- All? modern O/S have monitoring and diagnostic tools. We'll take a look at the one that comes with Windows.

Setting up perfmon

- Perfmon has one of the worst gui designs I have ever seen.
- Fortunately we don't have to use it.
- We use logman instead.
- Logman is the script interface to the perf-mon counters.

Register a counter set for logman

Typical command-line:

logman.exe create counter configname -cf d:\path\to\d\config.cfg -f csv --v -o d:\path\to\d\outputfile -y -si 1

These options mean:

Param	Description
-cf	Name of config file
-f csv	Use csv file format
--v	Attach file versioning information to the end of the log name
-o	Output path and file name
-y	Answer yes to all questions without prompting
-si 1	Sample interval in seconds.

What to log at disc level?

Counter	Description
\Cache\Data Maps/sec	Data Maps/sec is the frequency that a file system such as NTFS, maps a page of a file into the file system cache to read the page.
\LogicalDisk($DRIVE)\Disk Reads/sec	
\LogicalDisk($DRIVE)\Disk Read Bytes/sec	
\LogicalDisk($DRIVE)\Disk Writes/sec	
\LogicalDisk($DRIVE)\Disk Write Bytes/sec	
\LogicalDisk($DRIVE)\Avg. Disk Read Queue Length	
\LogicalDisk($DRIVE)\Avg. Disk Write Queue Length	
\LogicalDisk($DRIVE)\Current Disk Queue Length	is the number of requests outstanding on the disk at the time the performance data is collected. ... Requests experience delays proportional to the length of this queue minus the number of spindles on the disks. For good performance, this difference should average less than two.
\LogicalDisk($DRIVE)\Split IO/sec	reports the rate at which I/Os to the disk were split into multiple I/Os. A split I/O may result from requesting data of a size that is too large to fit into a single I/O or that the disk is fragmented.

- Use sed to convert $DRIVE to actual drive:
  sed s/\$DRIVE/d:\path\to\d\drive/g master.cfg > logman_drive.cfg

What to log at Firebird level

\Process(fb_inet_server#1)\% Processor Time
\Process(fb_inet_server#1)\IO Read Operations/sec
\Process(fb_inet_server#1)\IO Write Operations/sec
\Process(fb_inet_server#1)\Virtual Bytes
\Process(fb_inet_server#1)\Working Set

- Note syntax for fb_inet_server processes.
- You can log as many fb_inet_server processes as you want, even if they don't exist.

Start logman

Just use:

logman start counterset

Logman – useful things to know

- ***logman -?*** prints the help screen
- ***logman query*** will list all known configs and show their status.
- ***logman stop configname*** will stop a running config. Useful if you kill a batch file. Otherwise the dataset just keeps increasing.
- ***logman delete configname*** will do exactly what it says on the tin.

OK, we've got some data. Now What?

- That is a good question.
- Step One – analyse it for anomalies. Data caches are usually the problem.
- Once all caching is disabled we can start to run meaningful tests to compare, say, a single user with ten concurrent users.

Estimating required IOPS

RPM	Avg Latency	Avg Read Seek	RIOPS	Avg Write Seek	WIOPS
5,400	5.56	9.40	66	10.50	62
7,200	4.17	8.50	79	9.50	73
10,000	3.00	3.80	147	4.40	135
15,000	2.00	3.50	182	4.00	167

With the table above we can start to guestimate our requirements with one of the following:

POTENTIAL_WIOPS = NUM_DISKS * DISK_WIOPS

or

POTENTIAL_RIOPS = NUM_DISKS * DISK_RIOPS

Don't forget the write penalty

- If the test envionment is using a different disc sub-system to the target production environment then be sure to account for the RAID write penalty.
- You need to look at the split between WIOPS and RIOPS and adjust accordingly.

Summary

- We've looked at two different ways of measuring Firebird disc I/O.
- We've seen some of the caveats required when studying the data results.
- We've made a start at estimating disc sub-system requirements for Firebird.