Firebird performance degradation: tests, myths and truth

Check the original location of this article for updates of this article:
http://ib-aid.com/en/articles/firebird-performance-degradation-tests-myths-and-truth/

Recently in IBSurgeon we made series of performance tests with Firebird 2.5.2. Firebird 2.5.2 is the most popular version of Firebird database, and its users often have questions, related with Firebird performance.

One of the most important concerns regarding database performance is its degradation. Many users claim that their database applications have performance issues when reaching some threshold value: it can be 3Gb, 5Gb, size of RAM, 20Gb, etc. The most popular claim is “database size is more than RAM size”: when Firebird database size reaches the RAM size, it is reported to become very slow… Is it true?

We decided to perform series of tests to check is there such performance degradation related with database growth. We decided to simulate the lifetime growth of the database with the same load on the same hardware, and for this we have ran 11 tests with the databases of the size between 9Gb and 30Gb:

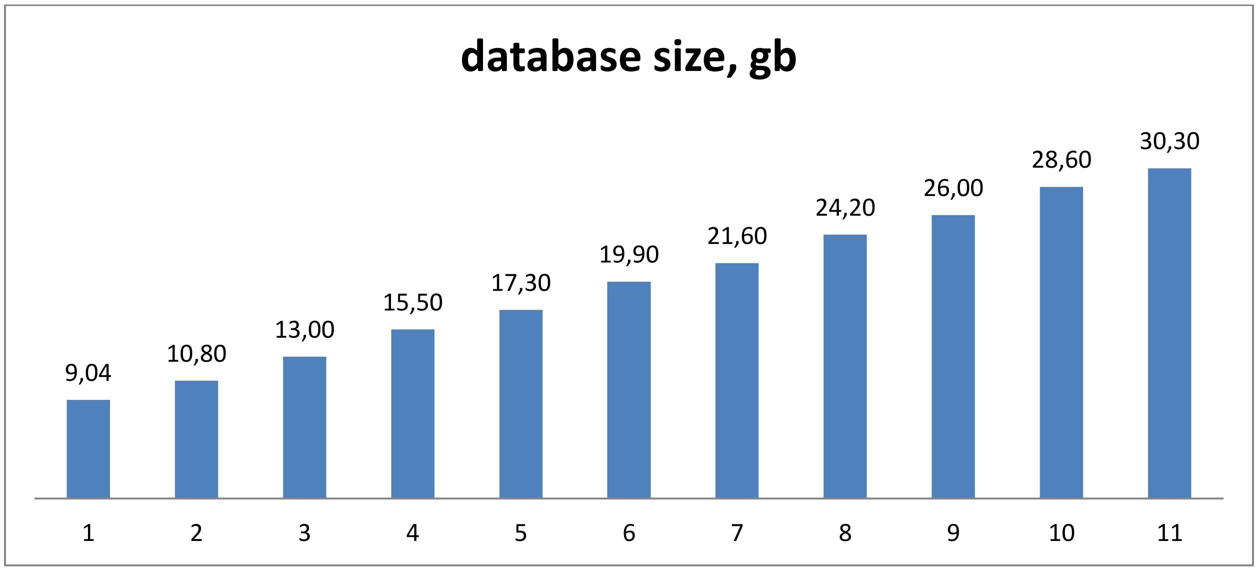


Figure 1 Database sizes for tests

Test hardware and Firebird configuration

Test hardware had the following key characteristics:

CPU AMD-FX8350, RAM 16GB, SATA software RAID1 2x4Tb Seagate drives, Operation System Windows Server 2008R2 (2008R2 is 64 bit).

As you can see, it’s a low-end hardware configuration; it can be bought for less than USD 1000 at the moment (May 2014), and it can be considered as typical low-level configuration - may be, except the large SATA drives, but according to manufacture report, speed of 1Tb and 4Tb SATA drives are almost the same.

Since the goal of the test was to measure performance changes of the typical system, we have used **Firebird (64 bit)** with **SuperServer** architecture, not Classic, to completely simulate situation in the small company – they use what was originally installed for years. As you know, SuperServer uses only 1 CPU core, so probably Classic or SuperClassic (which can use all CPU cores) could show the better results in terms of performance, but our goal was not the performance tuning.

However, we have tuned firebird.conf with the obvious changes we recommend to all Firebird SuperServer installations: increased page buffers to 10000 and temp space for sorting.

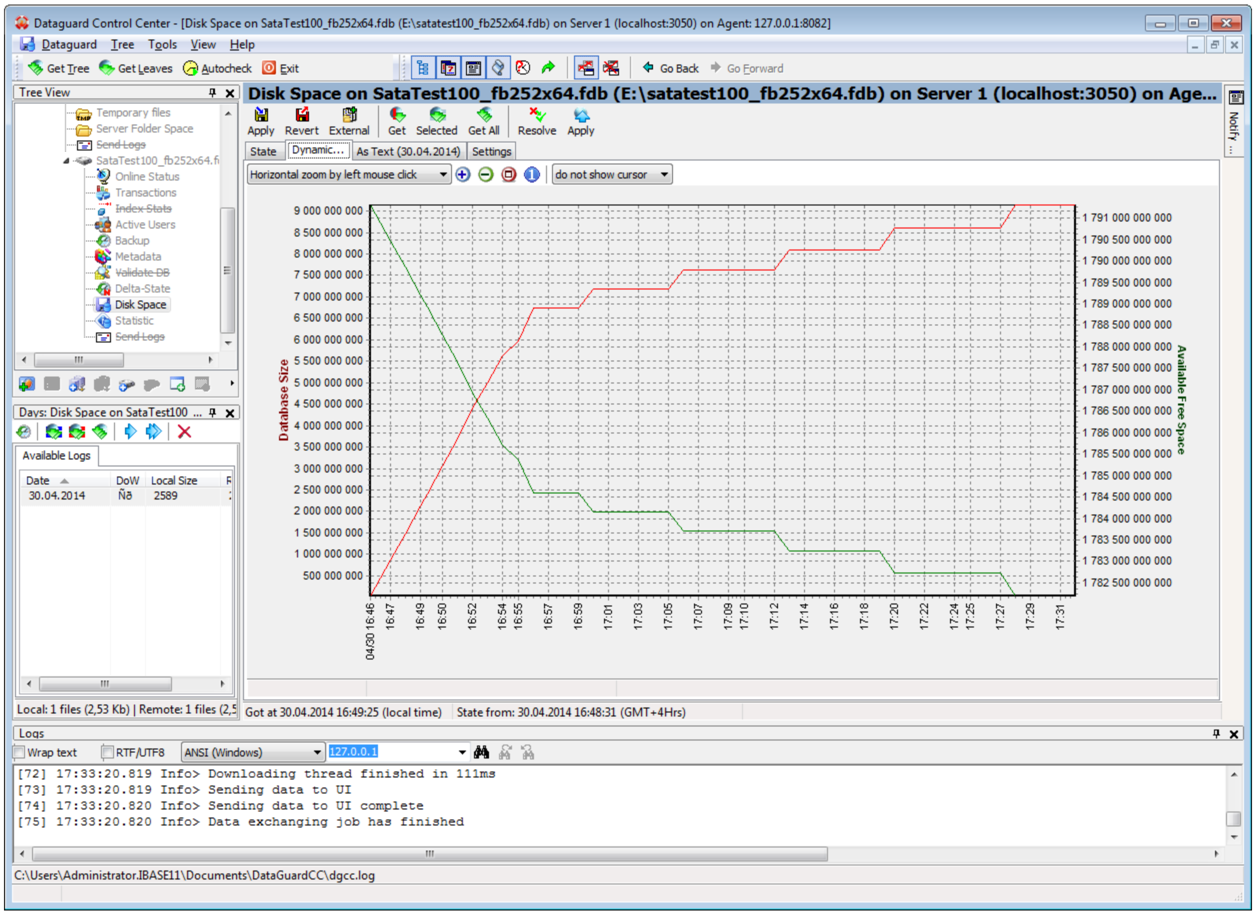
All test databases were created with page size 16384, just for consistency.

Testing

Loading

Each test contained 2 steps: loading and simulation of 20 terminals which perform inserts, updates and deletes.

Loading step is performed by loader application (load.exe), which inserts data into several tables. As you can see on the figure 2, data are loaded with different speed; it varies from ~35Mb/sec to 1mb/sec.


Dataguard Control Center - [Disk Space on SataTest100_fb252x64.fdb (E:\satatest100_fb252x64.fdb) on Server 1 (localhost:3050) on Agent: 127.0.0.1:8082]
Disk Space on SataTest100_fb252x64.fdb (E:\satatest100_fb252x64.fdb) on Server 1 (localhost:3050) on Age...
Database Size
Available Free Space
9 000 000 000
500 000 000
1 791 000 000 000
1 782 500 000 000
04/30 16:46
17:31
Got at 30.04.2014 16:49:25 (local time)
State from: 30.04.2014 16:48:31 (GMT+4Hrs)

```
[72] 17:33:20.819 Info> Downloading thread finished in 111ms
[73] 17:33:20.819 Info> Sending data to UI
[74] 17:33:20.820 Info> Sending data to UI complete
[75] 17:33:20.820 Info> Data exchanging job has finished
```

C:\Users\Administrator.IBASE11\Documents\DataGuardCC\dgcc.log

Figure 2. Database loading speed (red graph)

This is related with design of loader application, not with Firebird: loader quickly inserts 70% of the database and then slowly fills the rest of data, and it is repeated with databases of all sizes. It is important for us that loader performs the same operations, so we can use its average speed to measure loading speed.

It is important to say that loader inserts only data, and indices are created after load is complete.

Let's look at the table with results for loading step for 11 databases between 9 and 30Gb:

#	database size, gb	load time, sec	SATA load speed, Mb/sec
1	9,04	2535	3,65166075

2	10,80	3197	3,45924304
3	13,00	4057	3,281242297
4	15,50	4698	3,378458919
5	17,30	5455	3,24751604
6	19,90	6037	3,375451383
7	21,60	6473	3,417024564
8	24,20	7539	3,287014193
9	26,00	7779	3,422547885
10	28,60	8851	3,308823862
11	30,30	9266	3,348499892

Figure 3 Loading time and speed

Or, it is better to show on the graph at figure 4:

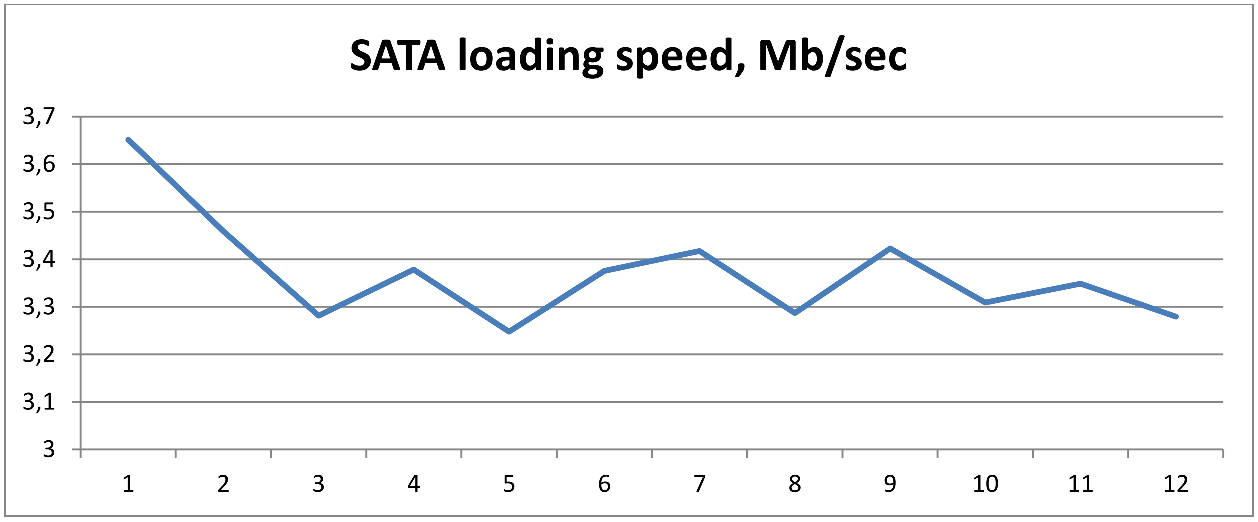


Figure 4 Test results: loading speed

As you can see, there is pretty stable graph, and average speed for the loading process varies around 3.3-3.4Mb/sec. There is also no signs of decreasing loading speed when database size becomes more than RAM size (after database #5, with size 17.3Gb).

Performance

So, loading times looks pretty promising, what about actual performance results?

Before going to the performance results, let's quickly review the simulation process.

The simulation runs 20 threads, and each of them randomly runs several business operations: create new order, process payment, count products in stock, process order delivery, etc (for details you can look at SQL texts of actual stored procedures). As you can see, this is usual set of business operations of abstract inventory/sales application.

Test application measures number of business operations per second, and reports an average number. Of course, this number is an artificial parameter, but it is good enough for comparison.

Performance test results:

#	database size, gb	performance, SATA RAID1
1	9,04	494,73
2	10,80	491,94
3	13,00	480,36
4	15,50	469,11
5	17,30	446,42
6	19,90	431,61
7	21,60	426,85
8	24,20	424,5
9	26,00	414,04
10	28,60	409,14
11	30,30	407,97

Figure 5 Table with performance test results

Or, it is better to view results in the graphical representation:

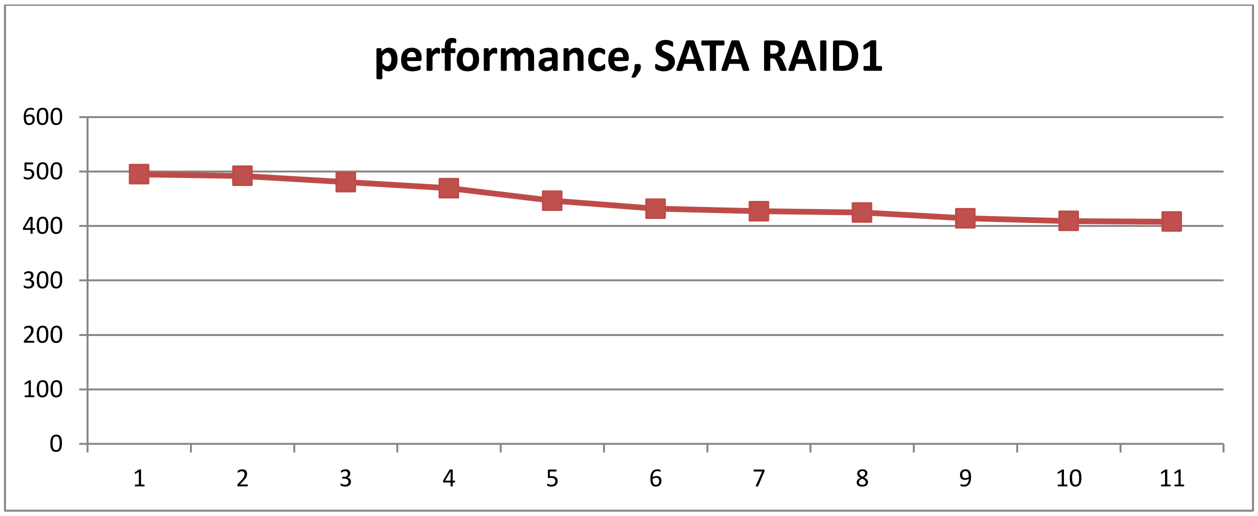


Figure 6 Performance results graph

As you can see, there is slow performance degradation with database size growth – the more database is, the slower it will work (on the same hardware). There is also no big drop of performance when database size becomes more than RAM. Database growth from 9Gb to 30Gb leads to approximately 20% loss of performance.

30Gb Firebird databases are all around now, and they grow and grow over time. However, what will happen with database performance when it will be even bigger? We mean – BIGGGER! What will happen with performance when database will become REAL BIG?

Mr. BIG

To answer this question we decided to look into the very end of test table and perform test with 1.7 Tb database, on the same hardware, with the same settings.

Loading

So, we created such database:

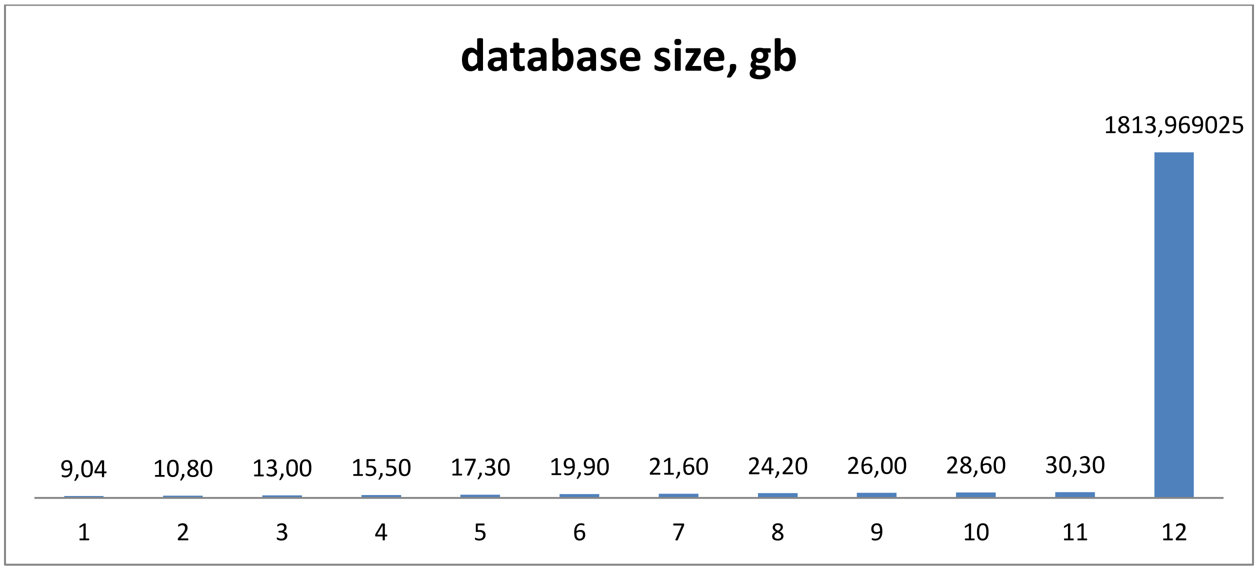


Figure 7 Database size - now with 1813 Gb database

Loading took 566448 seconds – 157 hours, 6.55 days. This is a long time, but average load speed was… 3.28Mb/sec!

Database size, Gb	Load time, sec	Load speed, Mb/sec
1813,969025	566448	3,279214122

Figure 8 Loading time and speed for 1.7Tb Firebird database

On the graph it looks very good: the last point (#12). So, Firebird shows very good results of its inserting algorithms.

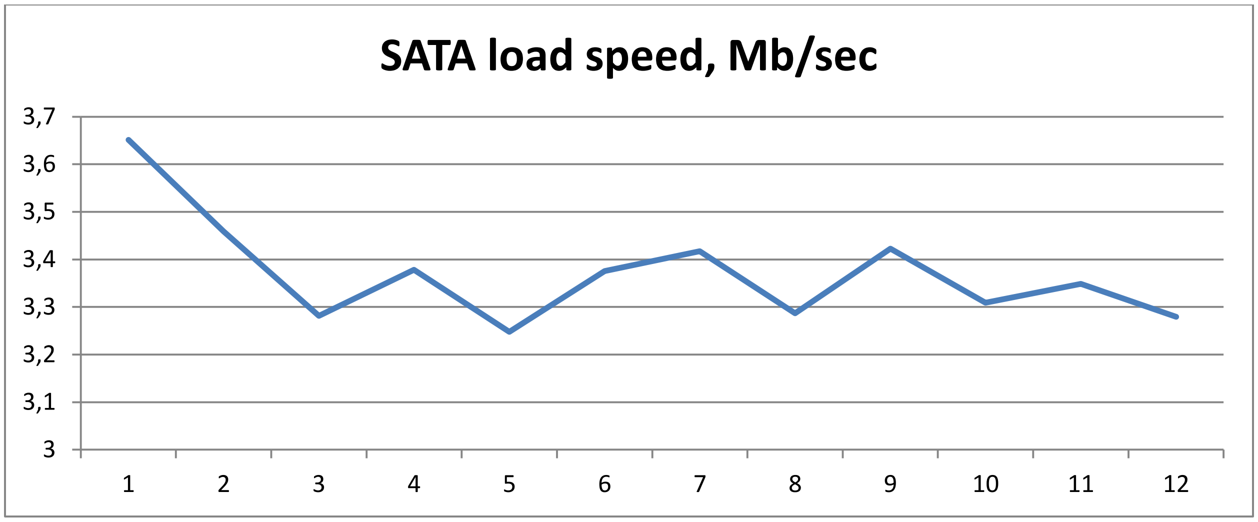


Figure 9 Loading speed – point #12 is for 1.7 Tb Firebird database

Performance of Mr.Big

Then we have run the same performance test – line 12 is for 1.7Tb database.

#	database size, gb	performance, SATA RAID1
1	9,04	494,73
2	10,80	491,94
3	13,00	480,36
4	15,50	469,11
5	17,30	446,42
6	19,90	431,61
7	21,60	426,85
8	24,20	424,5
9	26,00	414,04
10	28,60	409,14
11	30,30	407,97
12	**1813,969025**	**169,33**

Figure 10 Performance test results - line 12 is for 1.7 Tb database

And on the graph:

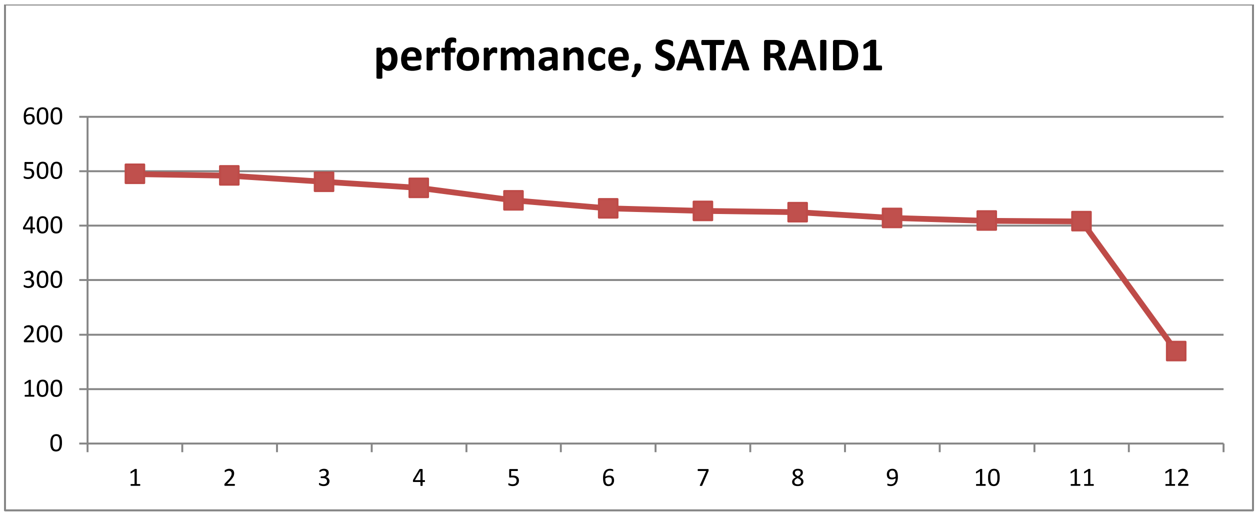


Figure 11 Performance, point #12 is for 1.7 Tb database

The result confirms that there is slow and stable performance degradation in Firebird – while the database size has grown 60 times (from 30Gb to 1813Gb), performance loss was 2.4 times (from 407 to 169 points).

It is not an often situation when database (on the same low-end hardware) grows from 30Gb to 1.7 Tb, but Firebird will work even in this situation.

Big database in details

To better understand the 1.7Tb database, we have gathered database statistics for Mr.Big database and analyzed in IBAnalyst:

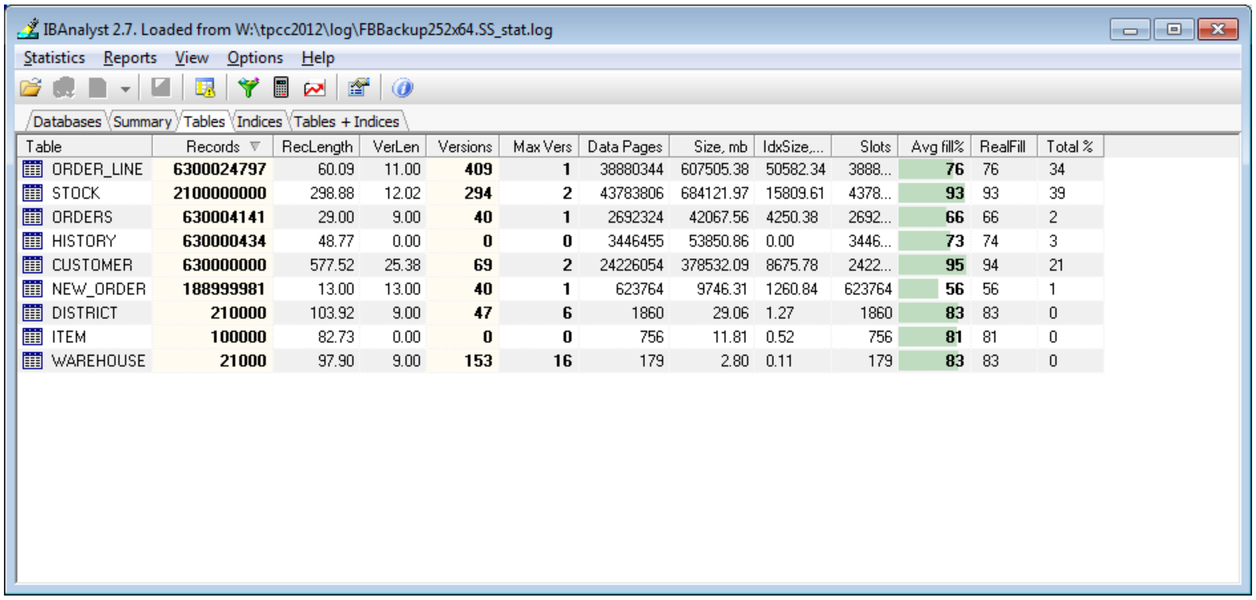
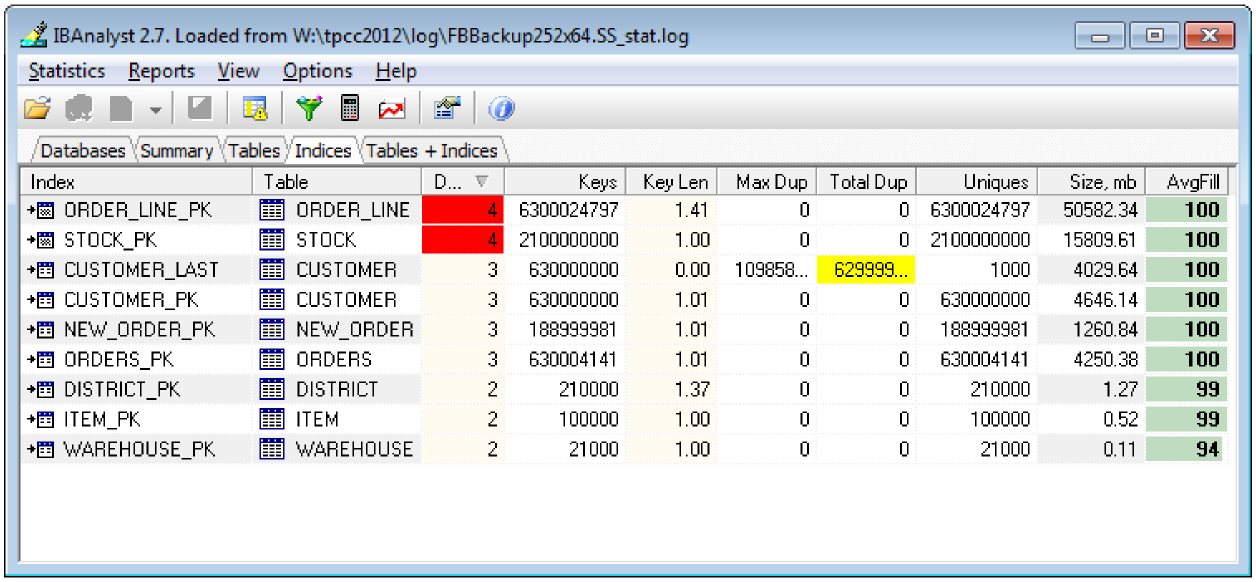

IBAnalyst 2.7. Loaded from W:\tpcc2012\log\FBBackup252x64.SS_stat.log
Statistics Reports View Options Help
Databases Summary Tables Indices Tables + Indices

Table	Records	RecLength	VerLen	Versions	Max Vers	Data Pages	Size, mb	IdxSize,...	Slots	Avg fill%	RealFill	Total %
ORDER_LINE	**6300024797**	60.09	11.00	**409**	**1**	38880344	607505.38	50582.34	3888...	**76**	76	34
STOCK	**2100000000**	298.88	12.02	**294**	**2**	43783806	684121.97	15809.61	4378...	**93**	93	39
ORDERS	**630004141**	29.00	9.00	**40**	**1**	2692324	42067.56	4250.38	2692...	**66**	66	2
HISTORY	**630000434**	48.77	0.00	**0**	**0**	3446455	53850.86	0.00	3446...	**73**	74	3
CUSTOMER	**630000000**	577.52	25.38	**69**	**2**	24226054	378532.09	8675.78	2422...	**95**	94	21
NEW_ORDER	**188999981**	13.00	13.00	**40**	**1**	623764	9746.31	1260.84	623764	**56**	56	1
DISTRICT	**210000**	103.92	9.00	**47**	**6**	1860	29.06	1.27	1860	**83**	83	0
ITEM	**100000**	82.73	0.00	**0**	**0**	756	11.81	0.52	756	**81**	81	0
WAREHOUSE	**21000**	97.90	9.00	**153**	**16**	179	2.80	0.11	179	**83**	83	0

Figure 12 Tables of 1.7Tb database in IBAnalyst

As you can see, there are 2 large tables – ORDER_LINE (~600 Gb) with 6.3 billion of records and STOCK (~GB680) with 2.1 billion records.

And, for this tables there are 2 indices with depth = 4 – it means that every request makes 4 reads of index pages before actual reading of the data. ORDER_LINE_PK index has 50Gb size.


IBAnalyst 2.7. Loaded from W:\tpcc2012\log\FBBackup252x64.SS_stat.log
Statistics Reports View Options Help
Databases Summary Tables Indices Tables + Indices

Index	Table	D...	Keys	Key Len	Max Dup	Total Dup	Uniques	Size, mb	AvgFill
ORDER_LINE_PK	ORDER_LINE	4	6300024797	1.41	0	0	6300024797	50582.34	100
STOCK_PK	STOCK	4	2100000000	1.00	0	0	2100000000	15809.61	100
CUSTOMER_LAST	CUSTOMER	3	630000000	0.00	109858...	629999...	1000	4029.64	100
CUSTOMER_PK	CUSTOMER	3	630000000	1.01	0	0	630000000	4646.14	100
NEW_ORDER_PK	NEW_ORDER	3	188999981	1.01	0	0	188999981	1260.84	100
ORDERS_PK	ORDERS	3	630004141	1.01	0	0	630004141	4250.38	100
DISTRICT_PK	DISTRICT	2	210000	1.37	0	0	210000	1.27	99
ITEM_PK	ITEM	2	100000	1.00	0	0	100000	0.52	99
WAREHOUSE_PK	WAREHOUSE	2	21000	1.00	0	0	21000	0.11	94

Figure 13 Indices of Firebird 1.7Tb database

Despite the huge number of records, database statistics looks good, so it's not surprising that Firebird shows pretty good results even for big database at low-end hardware.

Testing Firebird with SSD drive

After completing series of Firebird tests at low-end hardware we decided to check what will be results on another end of data storage technology and installed SSD drive on the same server.

We have installed SSD drive Plextor PX-256M M5 Pro, and run the same series of test (except 1.7Tb database), with the same settings. Results were added to the graphs with SATA devices, see them below.

Loading

As you can see, loading time at SSD is the same as SATA. This is an expected result: speed of sequential write operations is almost the same at SATA and SSD drives.

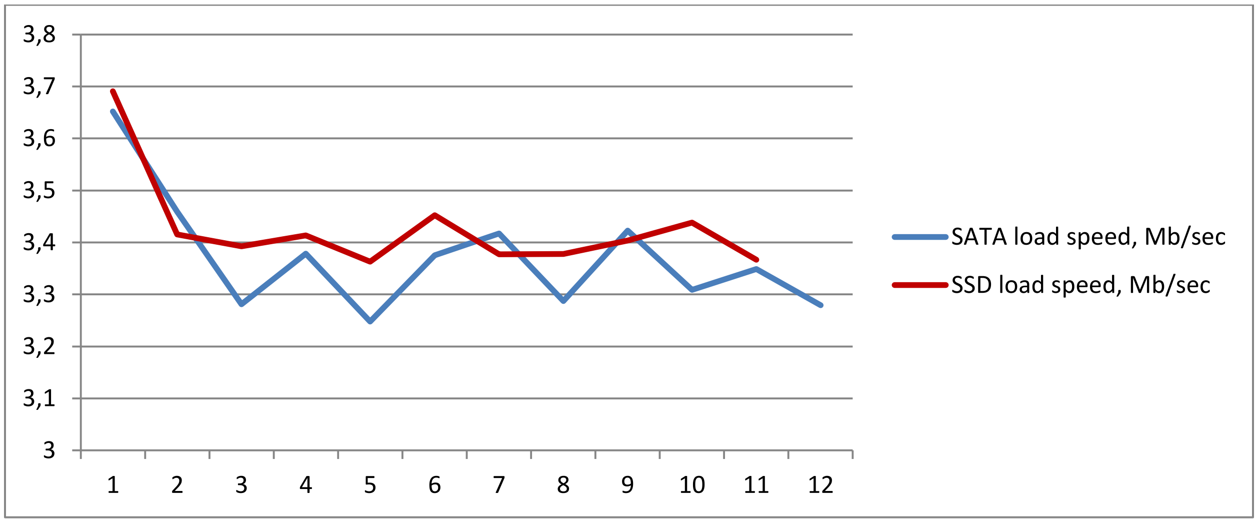


Figure 14 Loading at SSD and SATA

Performance

As you can see, performance with random IO operations shows ~8x better results for SSD drive. We knew from our experience with customers databases that SSD is 30-50% faster with real-word applications, but 8x increase is very high.

However, this test is artificial and specially designed to simulate high load OLTP operations, with many updates/deletes, but without large fetches. Usual database application does not work in this mode all the time. It explains why SSD shows such high results in this particular case.

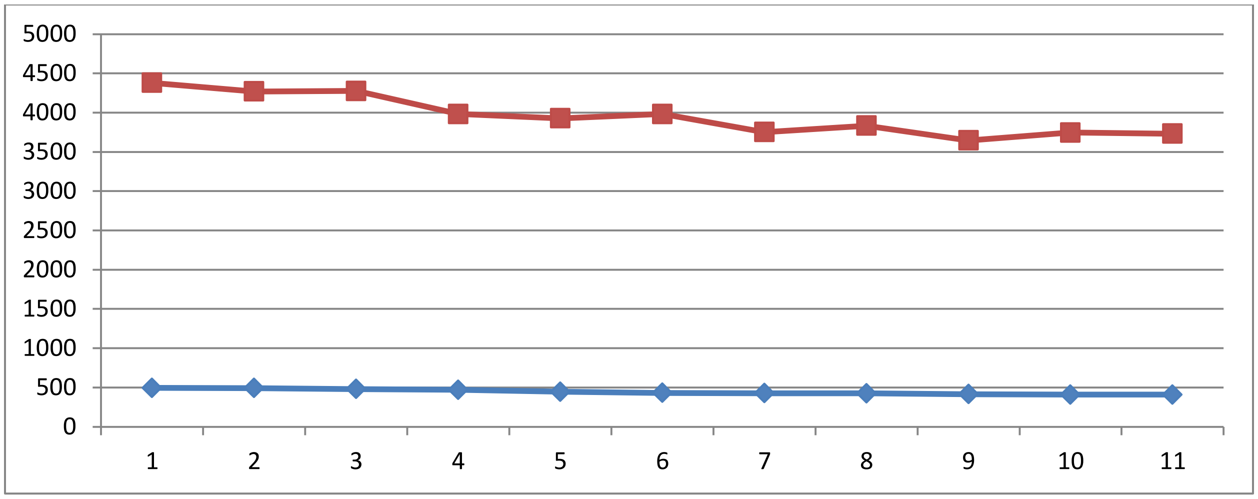


Figure 15 Perfromance at SSD and SATA

Summary

So, what we have learned from these tests?

First of all - performance of Firebird does not have big decreases related with some size restriction. On the same hardware performance will slowly decrease with the growth of database size. Such performance decrease can be compensated with Firebird configuration tuning or with smart hardware upgrade.

It's a good place to mention that IBSurgeon offers Firebird performance optimization service – using experimental data we gathered from tests like this we can significantly increase performance of Firebird and InterBase databases.

Then, we knew that even very large Firebird databases (1.7 terabytes) will work on the low-end hardware with significant, but acceptable performance loss.

And third, SSD is really good for OLTP applications. Probably it's the cheapest way to upgrade database performance at the moment. Of course, using SSD will not fix problems with bad query plans and ineffective indices, but it can raise performance in general.

Contact us

Please feel free to ask any questions: support@ib-aid.com