



Solid-State Disks: Sorting out the Myths From the Reality



By Michael Rosenblum

While testing solid-state disks (SSDs), it is important to ask the right questions. The main methodological approach for testers of SSDs is to distinguish between the overall performance of the system and the performance of the I/O subsystem. This article describes the results of an evaluation of SSD technology to see what performance gains can be achieved under different circumstances.



There have been many articles written about solid-state disks in the last few years. Those who have been looking at SSD technology seem to fall into two large groups. The first group believes that “It’s cool, but I don’t know how to use it.” The other group thinks that “It’s cool and can be used whenever I want.” This dichotomy is one of the reasons that solid-state disks are not as widely used as was predicted in articles written five to seven years ago. Each technology has its own limits, advantages and *Achilles’ heel*. Most manufacturers of solid-state disks are still explaining that their products are great. But why in some contexts are they are not so great? Why is the average system administrator still waiting for a miracle?

Setting up a Test

A few months ago Dulcian was able to run a simple experiment (thanks to Texas Memory Systems, Inc. for lending us RamSAN-220) to evaluate some possible ways of using SSD in the context of our Business Rule Information Manager (BRIM®). BRIM is an Oracle-based tool used to design and develop business rule-based enterprise information systems.

The testing was based upon the following assumptions:

- The SSD would be used by a mid-sized company running an Oracle database and Windows-based network.
- The company budget is limited so that the main consideration is not simply performance improvement, but performance/cost ratio.

- There is a significant learning curve. System administrators are not experienced in the use of SSD.
- There is not a lot of money in the budget to spend time testing the SSD technology.

Testing Environment

The following is a list of the hardware used to conduct the test.

SSD RAMSAN-220:

- 8 Gb SD-RAM
- Connected through one channel of HDA (Fibre Channel) from LSI Logic to the testing box #1

Test computer #1

- P4 1.7 GHz
- 1Gb RD-RAM
- 100Gb HDD – WD, connected using 100 Mb interface

Test computer #2

- Dual P3 600
- 1Gb SDRAM
- 4x18Gb SCSI Seagate Barracuda IV

Software list

- MS Windows 2000 Server SP2
- Oracle 9i Database Release 2 Enterprise Edition – pre-built configuration for general purposes. All database parameters are default.

I/O Testing

There is a lot of *common knowledge* about performance improvements, which can be gained about Oracle databases while using SSD (reallocating rollback segments, temporary tablespaces etc). These solutions are powerful, but very situation dependent. The goal is to find a generic analysis pattern and avoid standard solutions. A common situation is that an average System Administrator just received big new *box* with a lot of documentation. The documentation shows a few diagrams with some interesting data including the dependence of performance on block size. However, all of the databases are already set up for 8KB. This can easily be simulated using Intel IO Meter. Figure 1 shows a comparison of the block dependence claims of various manufacturers.

One channel connection is fairly close to the official performance numbers. A conclusion that can be drawn from the perspective of an Oracle DBA is that SSD is a promising place to put data that is accessed very often in decision support systems (where there are often larger data blocks). In these situations, SSD may dramatically increase total throughput.

The next logical question to ask is “Why bother?” What is the difference between SSD and other types of storage devices? In this case, the testing utility from SiSoft Sandra 2003 was used since it is generic enough as a starting point for this discovery process. Table 1 shows the results of the test.

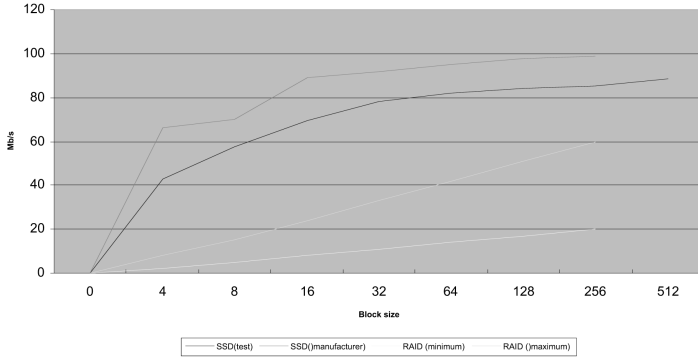


Figure 1. Block Dependence

These results are a little bit confusing. The close results in tests 1 and 4 can be explained by the extremely well optimized internal cache algorithms of the current hard drives, especially for reading. This test also indicates that the processing of streaming data is not the most powerful feature of SSD and that improvement of performance is inconsistent with the price paid for it.

Device	Test #1: Buffered Read	Test #2: Sequential Read	Test #3: Random Read	Test #4: Buffered Write	Test #5: Sequential Write	Test #6: Random Write
RamSan	95	98	98	86	84	82
IDE	85	40	6	65	38	11
SCSI	65	33	9	49	33	11

Table 1. SiSoft Sandra 2002 Test, Mb/sec.

Comparing tests #3 and #6, the difference is huge. However this is to be expected, since the ultra low latency period of the SSD not does differ in the method of storing data from the others. Further testing attempted to verify this concept using Intel IO Meter where the selected block size was 8KB (default block size for Oracle, which indeed had been used in the testing database), and the simulation was for 10 users. The test environment was limited to only SCSI and SSD. The results are shown in Table 2.

Device	Read		Write	
	Sequential	Random	Sequential	Random
SSD	87.0	85.7	58.7	57.7
SCSI	37.6	1.8	31.9	1.5

Table 2. IO Meter, Mb/sec.

In this case, SSD was clearly superior. Ten concurrent users made the SCSI hard drive option very slow, while the absence of physical movement in the SSD resulted in extremely fast response time per request and no delays between requests.

Conclusions From generic IO Testing:

SSDs vary in their advantages over other technologies depending upon which tasks they are used to accomplish. The highest possible relative performance improvement can be attained by using SSD for tasks requiring random access to stored data.

Oracle Testing

Using the previous test results, there is now a set of hypotheses that can be tested against a real Oracle database. To force the heaviest possible usage of SSD, it was mapped as a normal drive with a fully installed database. The testing hardware is the same box #1 (P4 1.7 GHz, 1 Gb RDRAM) with:

A. RAM-SAN 220

B. SCSI2 Seagate Barracuda IV

The database was populated with generic testing data. In addition, basic insert performance changes were also checked using the following commands:

```
create table a_perf (a number primary key,
                    b varchar2(200))
/
begin
  for i in 1..1000000
  loop
    insert into a_perf values (i,'Row#'||i);
  end loop;
  commit;
end;
```

A: 119.02 sec
B: 129.79 sec

These results were not very impressive. A 10% improvement can be gained by more common tuning procedures, but this exercise provided some testing data. To see whether or not the way in which data is being queried is relevant to performance (even access to the table will be by unique index), the same test was repeated with a slight modification to the UPDATE statement:

- 1) all records with a=i and i={1..100000}
- 2) all records with a=10*i and i={1..100000}
- 3) all records with a=100*i-j and i={1 t..100000} and j={1..10}

STATPACK snapshots were done before and after each test for future analysis.

Test	Test 1	Test 2	Test 3
Script	<pre>begin for i in 1..100000 loop update a_perf set a = a*(-1) where a=i; end loop; commit; end;</pre>	<pre>begin for i in 1..100000 loop update a_perf set a = a*10; end loop; commit; end;</pre>	<pre>begin for j in 1..10 loop for i in 1..100000 loop update a_perf set a = a*(-1) where a=(1)*(100*i-j); end loop; end loop; commit; end;</pre>

The idea behind this test was to determine the dependency of processing time on the physical location of the data. Since table *a_perf* is accessed by primary key *a*, this means that access occurs in the following sequence:

1. Search the index to find the ROWID of a column identified by the defined primary key.
2. Check to see whether this record is already in memory. (Oracle reads and writes by block, not by record so one block can have more than one record.)
3. If a record is found in memory (buffer cache), return it.
4. If a record is not found, access the hard drive using ROWID. This type of access is very precise, so only the needed block will be fetched. (In this example, there is only a small amount of data for each row, so row chaining did not have to be considered).

In Test 1, the program is accessing a contiguous set of data, assuming that multiple rows from the same data block are normal. It also assumes a minimization of read/write activities. Tests 2 and 3 should show how the performance will degrade when an increasing amount of blocks are processed. STATSPACK results are shown in the following three tables:

Test 1						
		A (RAM-SAN)			B (SCSI)	
Execution time	26.20			28.99		
Top Timed events	Waits	Time(s)	%Total Elapsed time	Waits	Time(s)	%Total Elapsed time
CPU time		24	93.19		24	78.67
log file parallel write	348	1	2.78	321	1	4.64
db file sequential read	531	0	1.65	384	2	7.60
db file parallel write	58	0	1.24	68	1	4.31

Test 2						
		A (RAM-SAN)			B (SCSI)	
Execution time	34.36			40.69		
Top 5 Timed events	Waits	Time(s)	%Total Elapsed time	Waits	Time(s)	%Total Elapsed time
CPU time		27	79.78		23	59.99
log file parallel write	517	3	5.14	311	1	3.27
db file sequential read	4984	3	10.25	4513	10	25.96
db file parallel write	144	1	2.94	144	3	7.71

Test 3						
		A (RAM-SAN)			B (SCSI)	
Execution time:	65.47			137.79		
Top Timed events	Waits	Time(s)	%Total Elapsed time	Waits	Time(s)	%Total Elapsed time
CPU time		34	45.81		33	19.37
log file parallel write	850	6	7.50	386	4	2.44
db file sequential read	57338	32	43.23	52691	102	59.94
db file parallel write	402	2	2.39	756	22	13.01

Based on these tables the following assumptions can be made:

- For the same task, the RAM-drive allows an increased percentage of CPU time. This means that less time is spent in I/O and the processor is used much more effectively.
- A database equipped with RAM-drive is very sensitive to the CPU/Memory subsystem. For the CPU-dependent tasks, the effectiveness of RAM-drive is low. Significant improvement can be achieved when the CPU time is much less than 50% of the total elapsed time.
- In general, the more data reading/writing required, the higher the speed that can be achieved using SSD. The implication here is that, before implementing an SSD-solution, it is strongly recommended that an analysis of what tasks will run on the database should be performed.
- Writing is always done in batch. *DB Writer (DBWR)* and *Log Writer (LGWR)* processes are not writing one record at a time. Depending upon the settings, the amount of writes is not directly dependent upon the number of rows. But reading happens per request, in this case, for each row. Some reading requests are processed by the cache, but in Test

3, the difference in writing is $26-8=18$, but in reading is $102-32=70$ for exactly the same 100,000 records. Therefore, applications with heavy querying demands can gain more from SSD than applications with heavy writing demands.

To test the first conclusion again, the previously mentioned BRIM® tool from Dulcian was used. This product has a large PL/SQL-based engine that is used for any DML-activity. The engine, called by triggers, processes each insert in its own way.

```
begin
  uml_global.setactiverule(1);
  security.connecttosystem('User','User');

  for i in 1..200
  loop
    insert into custmr (custmr_oid) values (i);
  end loop;
end;
```

A: 84.39
B: 95.24

This code creates new 200 new customers with predefined default attributes and associations, validates created values and processes each customer through the first few states of the State-Transition Engine. This represents a lot of work and showed insignificant improvement. Although 11% can be significant in large enterprise databases, this level can be reached by other common tuning procedures. If all tuning has already been done, SSD is an interesting way of accomplishing *the impossible*.

The last test involves merge joins which are extremely sensitive to the I/O subsystem because of their heavy usage of temporary tablespaces. This is a popular type of test so these results can be compared with data from James Morle¹. This case was very simple:

- 1000 departments with 500 employees each
- JOIN returns the pair department.name/employee.name.
- Merging is forced by hint /*+ USE_MERGE (department employee)*/

The results are impressive as shown here:

Test 1						
		A			B	
Execution time	9.53			29.15		
Top Timed events	Waits	Time(s)	%Total Elapsed time	Waits	Time(s)	%Total Elapsed time
CPU time		12	69.12		12	22.61
db file scattered read	1144	2	13.52	1147	15	28.91
direct path read	2155	1	6.60	2306	15	22.61
db file parallel write	504	1	3.81	512	4	4.89
control file sequential read	849	0	2.29	873	3	7.65

Our expectations were confirmed in that improvements were very significant. As mentioned earlier, CPU usage much below 50% for non-SSD usually guarantees extreme changes with RAM-drives. The relative change of speed in this test is higher than Morle's (67% vs. 25% and 379s to 281s). However, as mentioned in his article, the test was a "...quite CPU intensive operation." Once more, the conclusion can be drawn that the subsystem of the enterprise database with the lowest relative CPU-usage per task can achieve maximum performance improvement by implementing an SSD-based solution.

¹ *Solid-State Disks in an Oracle Environment: The New Rules of Deployment* James A. Morle, ScaleAbilities Ltd., 2002.

Summary

While testing solid-state disks, it is important to ask the right questions. The main methodological approach for testers of SSDs is to distinguish between the overall performance of the system and the performance of the I/O subsystem. In reviewing other Oracle tests, the improved performance due to SSD use was often neglected. The CPU can process only a certain number of operations per second. Therefore, CPU-consuming scripts cannot run faster even on the fastest storage device simply because they are not using it. The following are the important factors to analyze when selecting an SSD-solution:

- Streaming data flow/random data access
- CPU usage/I/O-activity
- Reading/writing

The conclusion that can be reached from the investigation described above is that SSD devices can greatly improve performance. However, there are many methods of doing this, not all of which are cost effective. Technologically, SSD is one of the best sources of performance improvement for an Oracle database assuming you have a typical OLTP system with a large number of users that processes many transactions accessing different small amounts of random data. SSDs may also improve data warehouse applications because of the improved query performance. There is no generic answer for all questions, but solid-state disks represent another way of thinking about managing enterprise-wide databases.



About the Author

Michael Rosenblum is a DBA at Dulcian, Inc. He also supports the Dulcian developers by writing complex PL/SQL and researching new features. Rosenblum can be reached via email at mrosenblum@dulcian.com.

RMOUG Training Days 2005

February 9–10
Colorado Convention Center
Denver, Colorado

Be a presenter at the best Oracle
user group conference in the country!

**Contribute your knowledge, share a useful tip,
be a part of the RMOUG tradition of excellence!**

If you are interested in speaking at RMOUG Training Days 2005, please submit your abstract via www.rmoug.org today! All primary authors of selected papers will receive a complimentary registration for the conference.

Suggested Topics:

- Application Design
- Application Development
- Data Warehousing
- Database Administration
- Network Administration
- Professional Development
- Technology Management
- Web, Internet/Intranet

Abstracts Due September 7, 2004. Submit Abstracts via www.rmoug.org.

