I/O Testing with Virident SCM

Using the Silly Little Oracle Benchmark

Martin Bach

5/8/2012

The following document describes read, write, and combined read/write tests using the Virident N800 SCM on a 2 socket 6 core Westmere EP server using Kevin Closson's Silly Little Oracle Benchmark.

SLOB TESTING WITH VIRIDENT FLASHMAX

As part of the same project I have been working on furiously recently I had the immense pleasure to test-drive a FlashMax PCIe card from Virident. Virident I hear you say, I have not heard about them yet! Tell me more.

When I approached them with an overview of what I wanted to do and why, they were very open and sent me access detail for a system with one of the single level cell FlashMax drives. That was until then the fastest server I connected to: 2 Westmere EP processors, Xeon 5670 @ 2.93 GHz. In total, I had a 2s12c24t, which reads 2 sockets, with 6 cores each for 12 cores in total. Since the 5670 is hyper-threaded it gives you 24 threads. Unlike some AMD 6100 and 6200 chips, AWR recognises the Intel package and reports it correctly. My many thanks go to Kevin Closson for introducing me to Virident.

Having recently been a beta-tester for SLOB (see http://bit.ly/zYq2mX), the silly little oracle benchmark I was very curious to see how it works with PCIe based cards. Obviously, it requires a powerful machine to saturate a card, but the server I was given was positively fast. My draft test plan included these:

- Install Oracle 11.2.0.3 Grid Infrastructure and Oracle 11.2.0.3 RDBMs software
- Create a database according to SLOB's cr_db.sql script, using the same minimalistic init.ora
- Run SLOB to ascertain physical I/O rates, redo sustained write capability and maybe logical IOs to round the picture up

FALLACIES AND PITFALLS

I really like the end of chapter section in "Computer Architecture: A Quantitative Approach" by John L. Hennessy, David A. Patterson. In it they present some examples of how NOT to do things. The same principle ("do not do the wrong thing") is true for testing any flash cards, regardless whether they are attached via a SATA/SAS ports on the board, via PCI Express, or external connectivity such as Fibre Channel and Infiniband.

First, it is important to clarify terminology. What is called a solid state disk or "SSD" in consumer terms is actually NAND flash. NAND has to do with the internal organisation of the chips, they are non-volatile (unlike DRAM based SSD) and you know them from the iPod or comparable device. There is DRAM based flash memory as well, but it has not found widespread adoption.

NAND flash comes in two flavours: one is the Single Level Cell ("SLC"), the other one is "Multi Level Cell". The simplified version of the truth is that SLC used to be more or less exclusively the domain of enterprise class flash, and it is fast. SLC store only 1 bit of information which means there is less data density. To compensate and compete with other storage solutions, MLC can store more information per cell. MLC was initially less accepted, but it allows for more capacity, at the expense of performance.

For the following paragraphs it is also important to know how data is organised in the SSD. The cells (MLC/SLC) are usually organised in 4k pages. Multiple pages form a block, which often is 512kb. Why does it matter? When the flash device is fresh/new from the factory it will be all empty-none of the cells contain any data. This makes it easy for the controller to write very quickly-it simply selects empty cells. When the flash device is filled, the free-list maintenance becomes trickier and eventually the controller cannot simply write, but it has to erase. This is an expensive operation, because an erase requires the whole block containing the modified page to be read: in the worst case you modify a 512k block to change a single page. So you read the entire block, erase/modify, and then write back. Clearly this has an effect of performance which vendors are normally not too keen on reporting. Unlike Virident. When I started testing I was advised that I should use dd to write data over the whole device to get more realistic performance data. Finally I found a vendor who cares more about engineering than marketing.

So the bottom line with any SSD related testing is that until the device is "full", there is a blindingly fast initial phase. Don't assume these numbers will be standard. Sadly many data sheets will claim the opposite.

The reason for the confidence in Virident's advice is simple: according to the engineer I spoke to the card is optimised such that it can keep its high write rates even in garbage collection mode, which is called "sustained" mode on their card.

Another advantage of the Virident card is its small form factor which makes it suitable for most deployments, even blades. The particular card I was given was a N800 which is an SLC drive with 800G usable flash with 2.1.2 GA software on it. There are others available with a capacity of up to 1.4 TB.

SLOB TESTING

The Silly Little Oracle Benchmark does not need to be introduced anymore. Written by Kevin Closson it is designed to perform serious I/O in various scenarios. It can be instructed to either read data from Oracle, or write data to it or both. When reading, the code randomly selects a row from the test data set, a write updates a random row. Since it doesn't matter what is read or updated (all we care about is the time that operation takes!) you don't have to re-generate the test data. A word of warning: don't run SLOB in production, it has the potential to seriously flood your array's front-end ports.

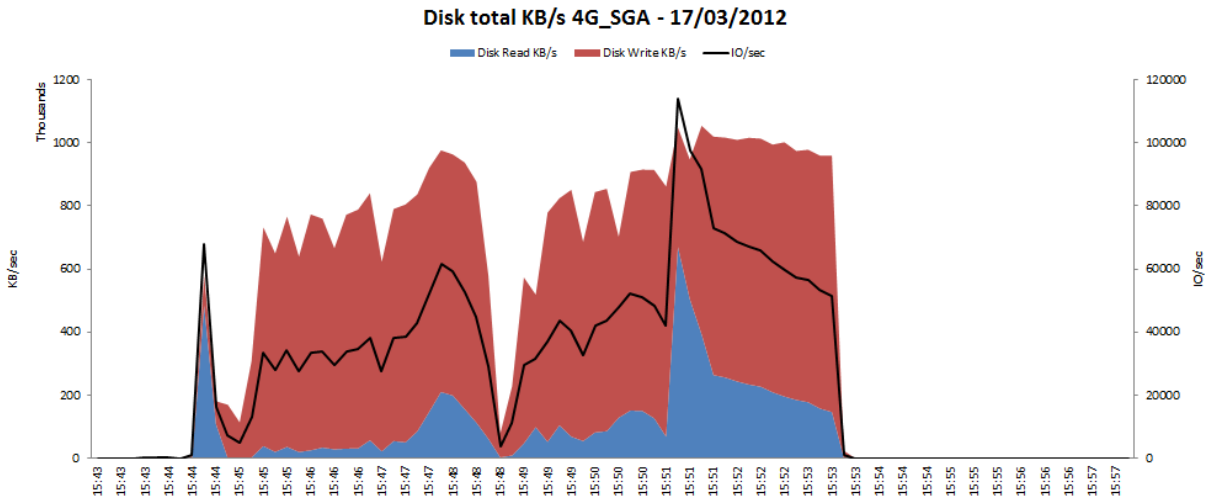
Frits Hoogland has recently done a great presentation about running SLOB on different hardware, and this analysis follows a similar approach. Before starting with the results, I'd like to present an overview of the hardware used.

The server was a 2 socket Intel Xeon X5670 clocked at 2.93GHz-Westmere EP, with hyper threading enabled. The operating system saw 2s12c24t. There were 24GB of RAM available, out of which I used 4 for sga_target to drive the below test. The FlashMax card was the single ASM disk in disk group DATA for an Oracle Restart installation.

The Linux distribution installed on the server was Red Hat Enterprise Linux 5.4 with kernel 2.6.18-164.el5. The scheduler used was the default for the card and the operating system, due to time constraints no experiments could be conducted with different schedulers. I also would have liked to experiment with kernel UEK but that is not available to Red Hat 5.4.

SYSTEM OVERVIEW

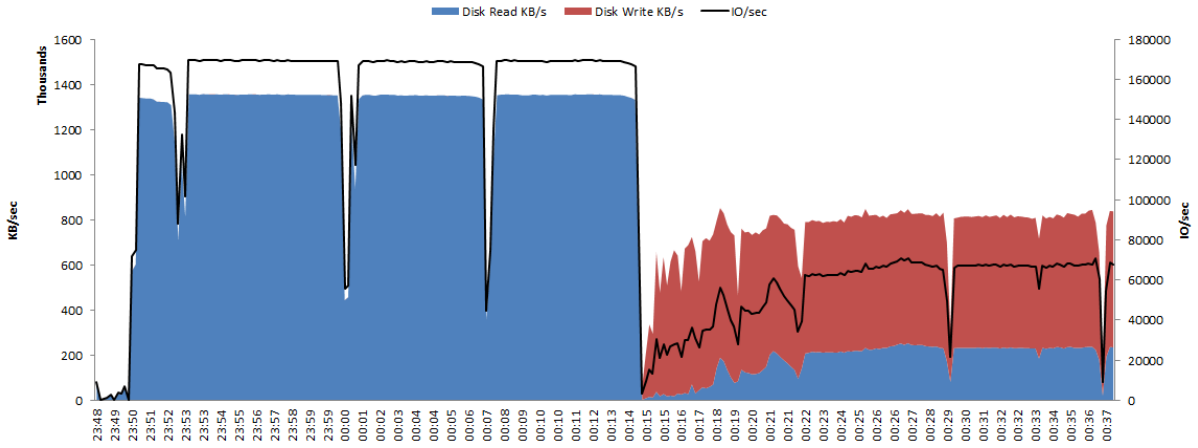
NMON reported the following profile during my initial test. My load profile started with 1-24 slaves for reads (no writers), followed by an inverse run of 1-24 writers with no readers. I then combined the two.



The dip in performance is due to the start of the second test series-there was no problem with the card.

As you can see the test case is able to generate 113775.8 IO/sec, with a maximum of 669912.1 kb read per second and 812412.3 written per second. At the peak-when nmon recorded 113775 IOPS those resulted in 1049188 kb/s to which 669912.1 ks reads/s and 379275.9 kb writes/s attributed.

Another test has been performed from which we gathered that 64 readers/64 writes saturate the system. The below is the nmon output from a brute force run with iterations of 0-512 readers and writers.

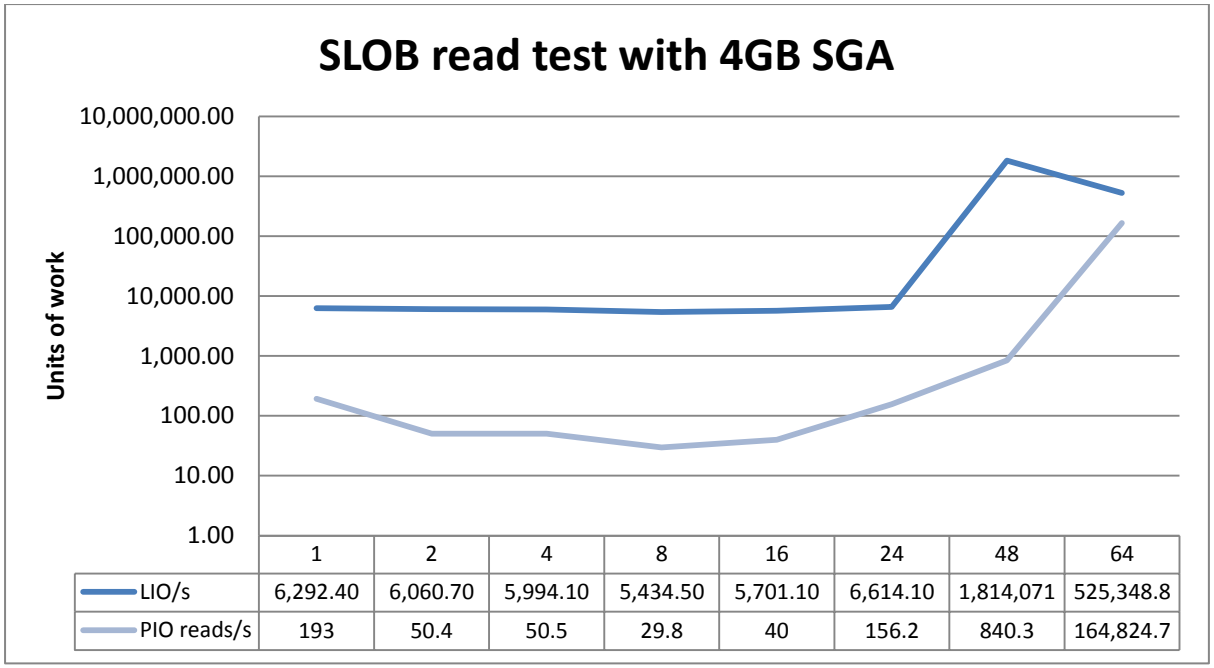


These figures indicate a maximum IOPS of 168997.3 to generate 1351975.9 kb read/s during the reader test. The peak for reads was 586326.1 kb written/s, 232419.1 kb read/s at 67169.3 IOPS. Again, the dip in the centre is the change from reader to writer test, when the next series of tests commenced.

TESTING READS

All the read-tests I conducted involved iterations of runit.sh for 1-64 readers. My SGA was set to a constant of 4GB, using sga_target and automatic SGA management to that effect. I chose this approach to compare results I received from other SLOB users.

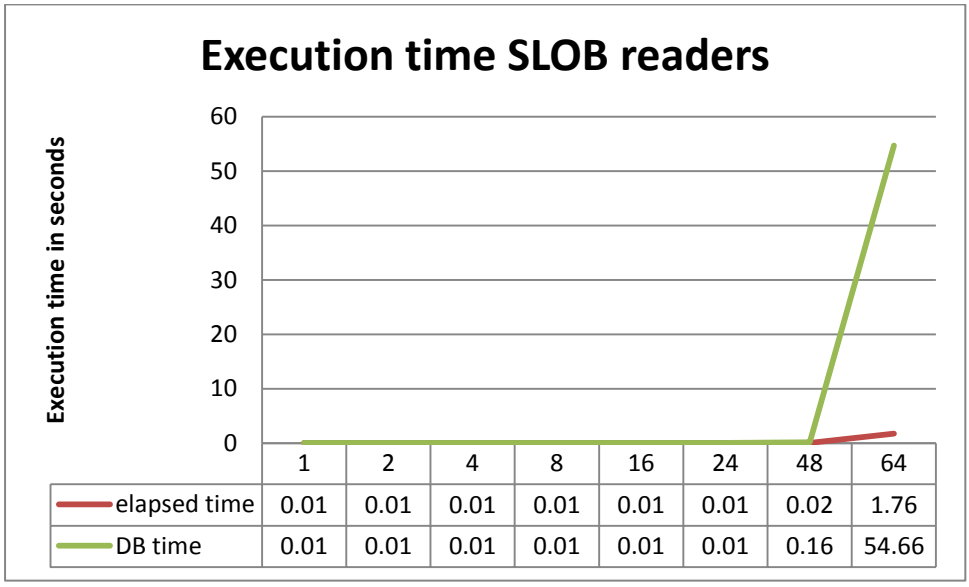
The below is a summary of the information presented in the AWR reports generated by each execution of runit.sh.



	1	2	4	8	16	24	48	64
LIO/s	6,292.40	6,060.70	5,994.10	5,434.50	5,701.10	6,614.10	1,814,071	525,348.8
PIO reads/s	193	50.4	50.5	29.8	40	156.2	840.3	164,824.7

Note that the y-axis is of a logarithmic scale to base 10 to fit on screen. Also you can see the data points for each of the runit.sh invocation-the x axis shows the number of readers. The number of logical reads/s stays fairly constant up to 24 readers and then shoots up. This is remarkable for me as it exceeds the number of threads available to the system. It drops when the number of readers reached 64, which saturated the FlashMax card. However, the number of physical reads shot up to a maximum of 164824.7.

What's not reported here is execution time. I have seen many systems that required a long time to execute runit.sh with > (thread or core count) * 2, but have a look at these:



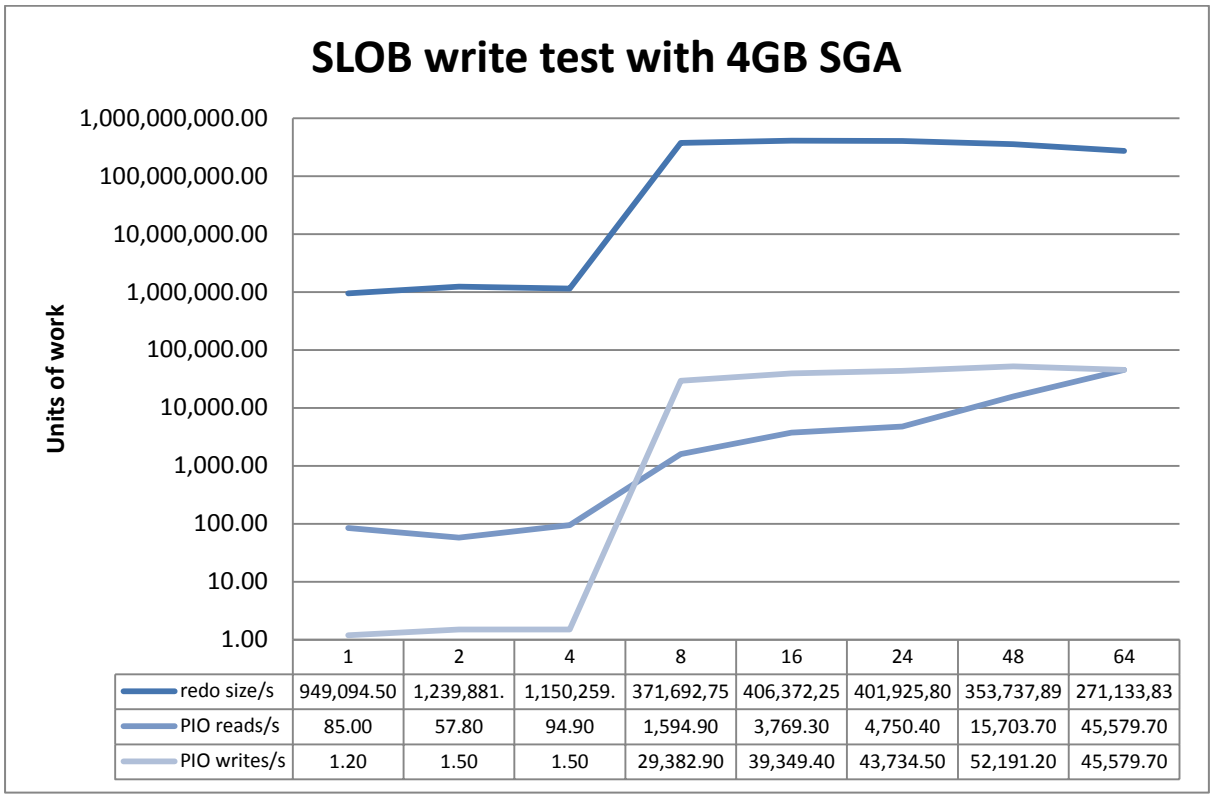
	1	2	4	8	16	24	48	64
elapsed time	0.01	0.01	0.01	0.01	0.01	0.01	0.02	1.76
DB time	0.01	0.01	0.01	0.01	0.01	0.01	0.16	54.66

The red line is execution time as measured by the wall clock. AWR reports it as 1 second, rounded up, except for the 64 readers-case where it is 1.76 seconds. The DB time in the last scenario went up to 31 seconds. Try this with your storage!

TESTING WRITES

The pfile remained the same to allow for comparable results during tests. Again, the sga_target was set to 4GB without lower limits for any of the automatically managed components.

The results were equally impressive, as shown in the below figure.



	1	2	4	8	16	24	48	64
redo size/s	949,094.50	1,239,881.	1,150,259.	371,692,75	406,372,25	401,925,80	353,737,89	271,133,83
PIO reads/s	85.00	57.80	94.90	1,594.90	3,769.30	4,750.40	15,703.70	45,579.70
PIO writes/s	1.20	1.50	1.50	29,382.90	39,349.40	43,734.50	52,191.20	45,579.70

Again I had to opt for a logarithmic y-axis to fit all data on the chart.

SUMMARY

I am very grateful for Virident to have given me the opportunity to use their Flash Max card and the server. During background reading I found a lot more information about the card and its architecture plus some benchmark results of a combination of multiple cards in a NEC server (7U!) that delivered insanely high IOPS with great sustained throughput. The cards are able to deliver even higher throughput at 4k block size, but to keep the research in line with my other material I deliberately decided to use 8k as the great majority of database systems does.

As you can see the number of IOPS is clearly very high, and the throughput isn't bad either. There were no inacceptable IO latencies during the test, most of which completed very quickly. Remember though that SLOB uses db_block_size I/O request, so when running collectl and viewing the IO sizes you won't see large numbers.

ABOUT VIRIDENT

If you are curious who Virident are, here's the statement from their official LinkedIn profile:

Founded in 2006, Virident Systems helps enterprises solve the challenge of poor application performance with SCM (storage class memory-ed) flash storage solutions that speed application performance, ensure predictable performance under various workloads, and are compatible with all servers.

Virident FlashMAX, a PCIe storage class memory (SCM) solution, ensures tenfold improvement in application performance at one-third the total cost of ownership of HDD systems. Virident vFAS (Flash-management with Adaptive Scheduling Layer) software stack, tightly integrated with FlashMAX , delivers unconditional performance that scales across diverse workloads and data sets, and remains consistent over time. It also enables maximum capacity utilization resulting in at least twice the price/performance of comparable flash-based solutions. Built-in flash-aware RAID ensures the highest enterprise-class reliability and data availability. Virident FlashMAX is optimized for demanding storage environments in the energy, government, finance, manufacturing, digital media and Web 2.0 industries. Virident Systems is backed by Sequoia Capital, Globespan Capital Partners, and Artiman Ventures.

Virident Systems was founded by a visionary management team from Google, Sun Microsystems, Cisco, SGI, and Intel. The privately held company is backed by Cisco, Intel, Sequoia Capital, Globespan Capital Partners, and Artiman Ventures.