



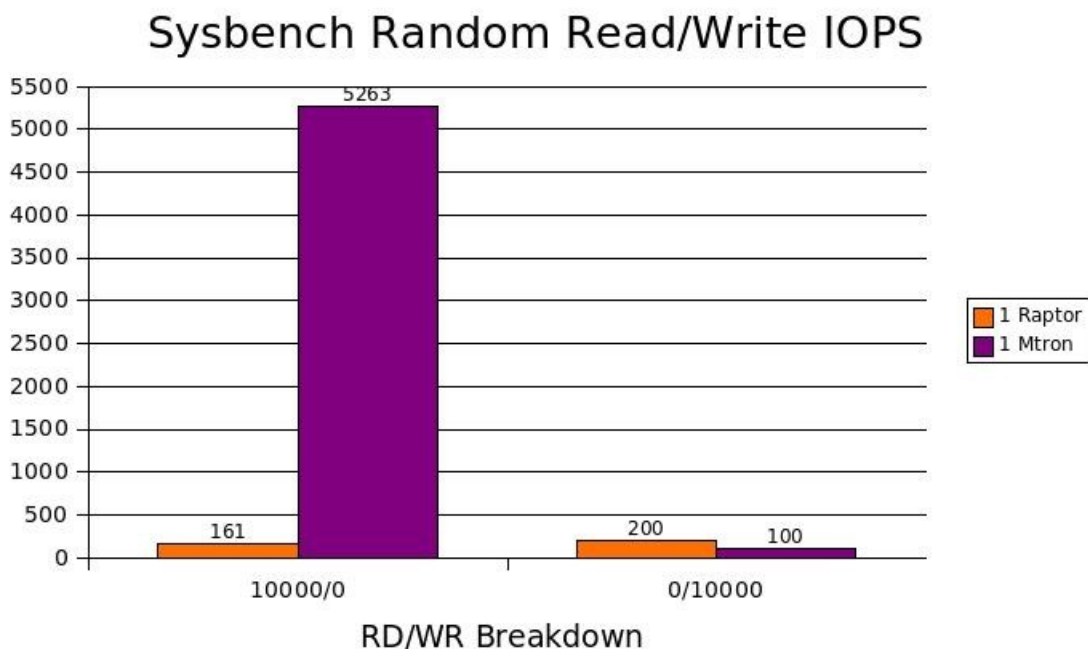
Solid State Disk Benchmarking

Matt Yonkovit
Senior Consultant, MySQL AB

This document shows performance benchmark tests using a 32GB Mtron MSP Solid State Disk Drive in a Linux Environment. The focus of these tests is to show the potential performance boost for running MySQL on one of these drives. In addition to testing the performance of the standalone SSD drives I also tested a driver from Easy Computing Company called MFT that enhances the write capability of these SSD drives. Excerpts below were pulled from my Blog.

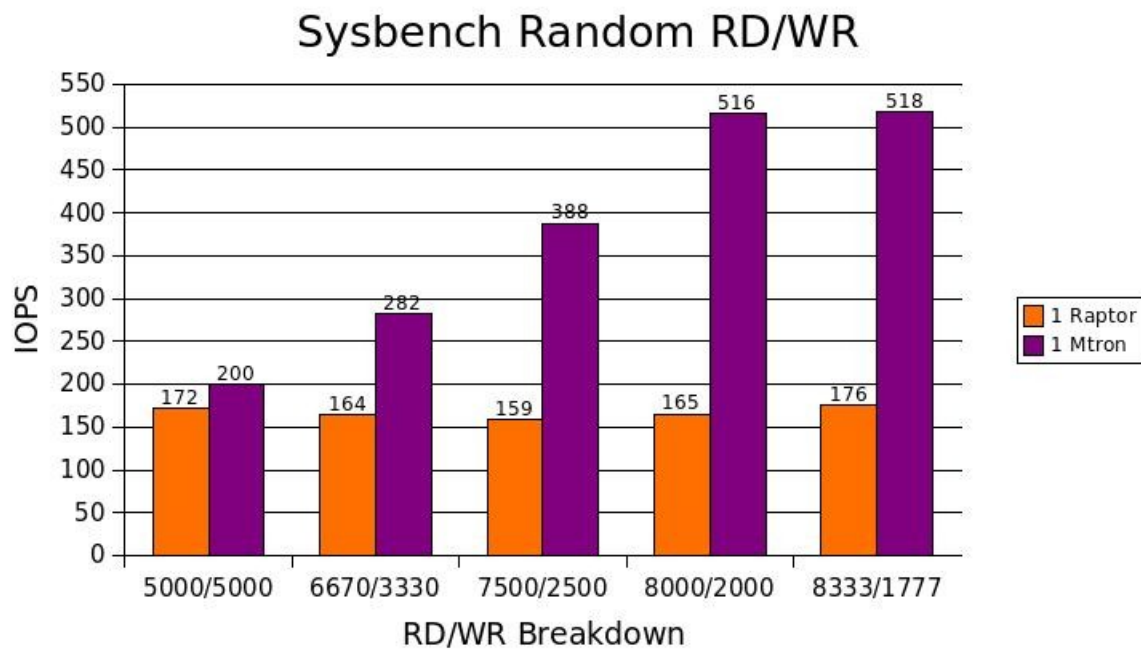
The test setup is 1 4 Core Centos 5.1 machine with 6GB of Ram. The controller used is a standard 8 port SATA Supermicro controller.

The performance of these drives really shines with a specific workload, but they are not for everyone out there. The random write performance of these drives leaves a great deal to be desired while their read performance is outstanding:



Above you can see that when we perform 10K random reads with 0 writes we peak at about 5200 IOPS vs the 161 IOPS on a standard SATA drive. When we flip the IO to all writes we end up getting around 100 IOPS out of the SSD drive. Not many sites are 100% reads, so some sort of mixed IO load is

expected. Here you can see how the number of IO's per second varies under different workloads:



I have 2 drives, I will rerun these in RAID 1 & RAID 0 to see how things change. But based on the above sys bench tests in a 80-20 read/write environment the Mtron does show a 3X increase. What is a little odd is when I moved to OLTP testing. I used 100Millions rows with 32 threads. According to the raw output from the test there are about 70% reads in the OLTP test, see below:

Maximum number of requests for OLTP test is limited to 50000

Threads started!

Done.

OLTP test statistics:

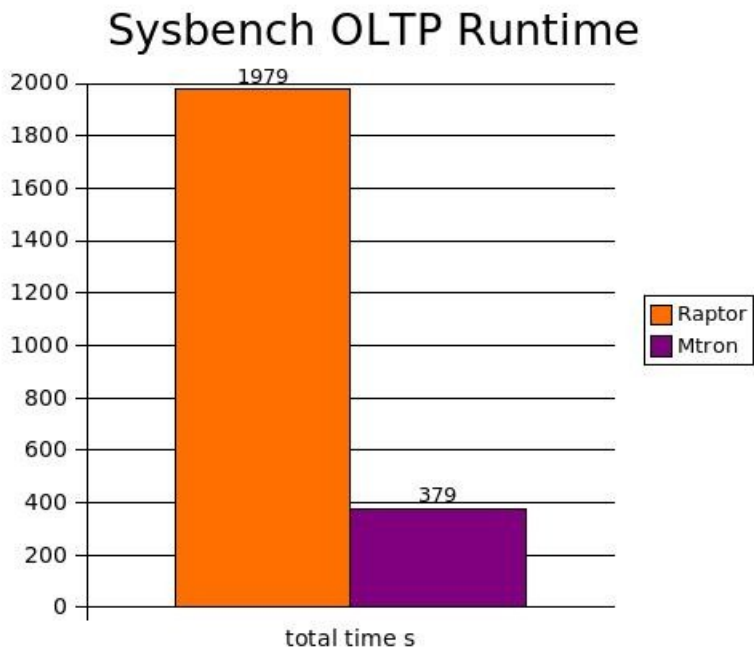
queries performed:

read:	700098
write:	250035
other:	100014
total:	1050147
transactions:	50007 (131.93 per sec.)
deadlocks:	0 (0.00 per sec.)
read/write requests:	950133 (2506.74 per sec.)
other operations:	100014 (263.87 per sec.)

Test execution summary:

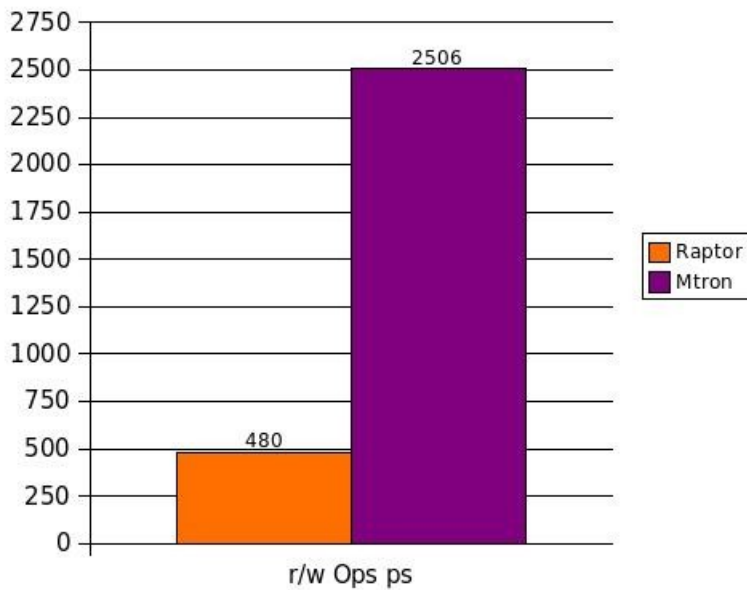
total time: 379.0310s
total number of events: 50007
total time taken by event execution: 12126.7578
per-request statistics:
min: 0.0028s
avg: 0.2425s
max: 2.3525s
approx. 95 percentile: 0.7041s
Threads fairness:
events (avg/stddev): 1562.7188/19.88
execution time (avg/stddev): 378.9612/0.02

The results I received show a much higher performance benefit than the raw sysbench IO tests:

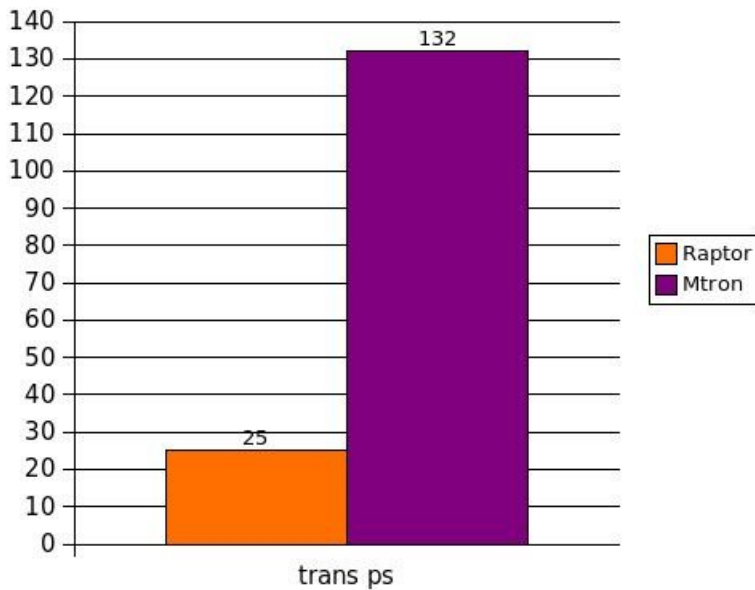


Run time dropped from 1979 seconds on a single raptor drive to 379 Seconds on the standalone Mtron drive. An improvement of over 5X. Based on the generic disk benchmarks I would have thought I would have seen slightly lower runtimes. The 5X seems to hold up looking at the other numbers as well:

Sysbench OLTP OPS PS



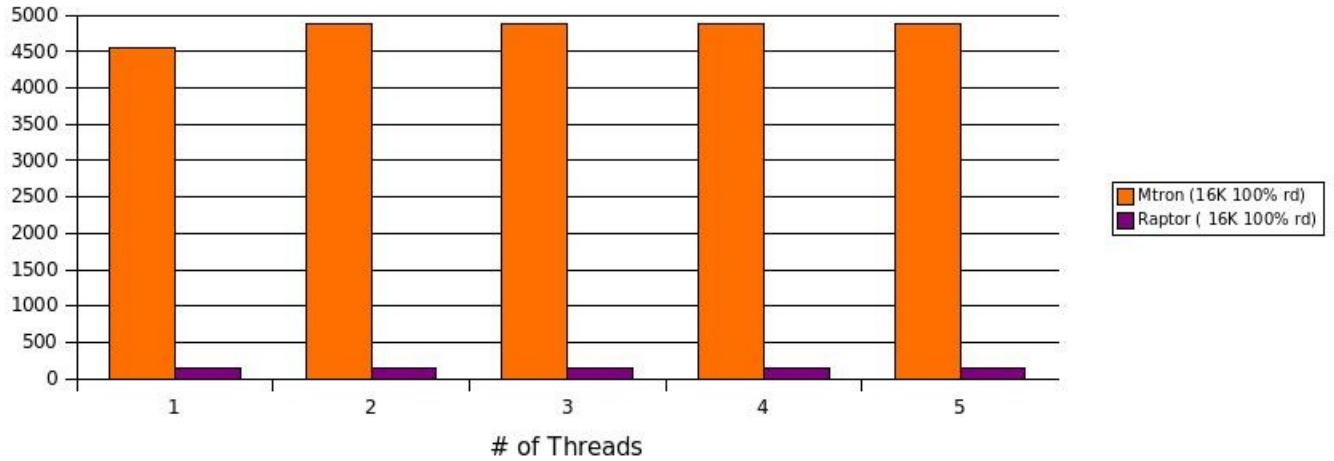
Sysbench OLTP Trans PS



On top of Sysbench I also ran [Orion](#). Orion is an oracle disk benchmark tool that they released several years ago. It is freely available and does a great job of benchmarking disks, breaking down performance based on the # of threads hitting the disk and showing latency at each point. Lets take a look at the numbers.

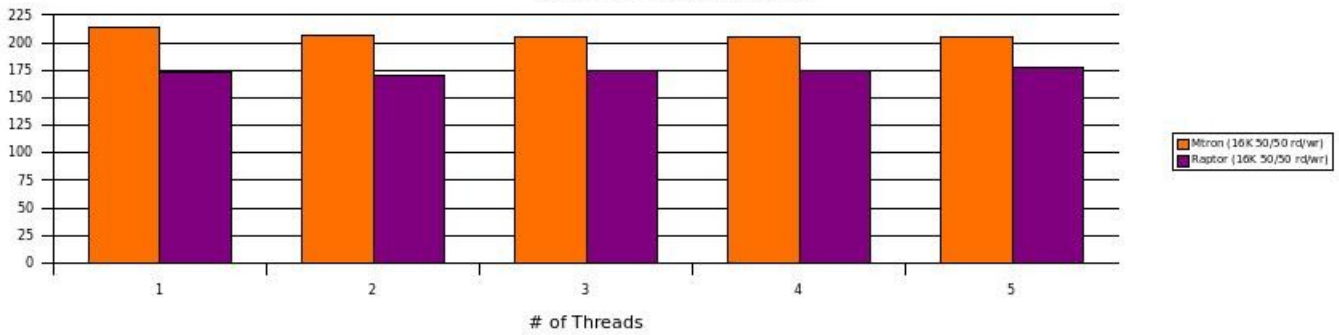
We already established reads were screaming, but lets check Orion just to help validate things:

Orion IOPS 100% Reads



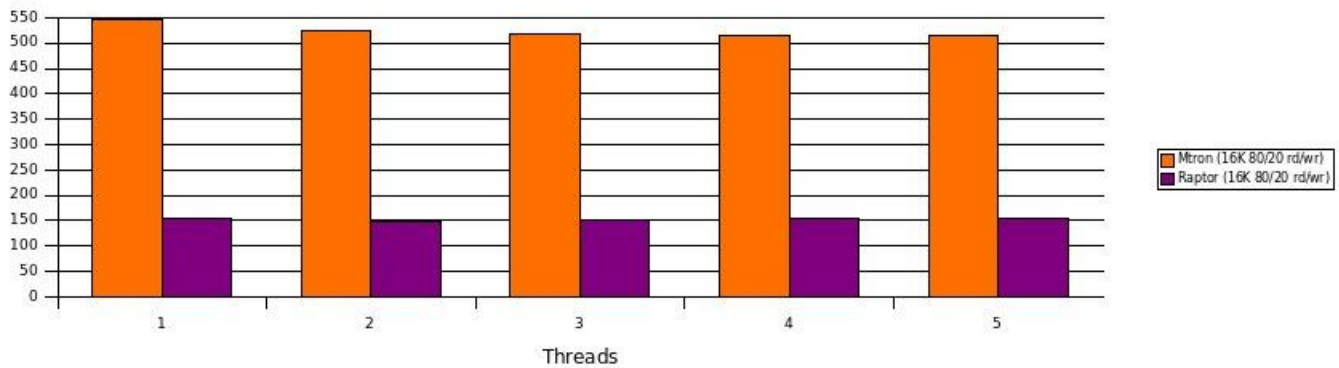
Same thing... nothing more to say about that. Now lets look at the 50% read/50% write environment:

Orion IOPS 50% Writes

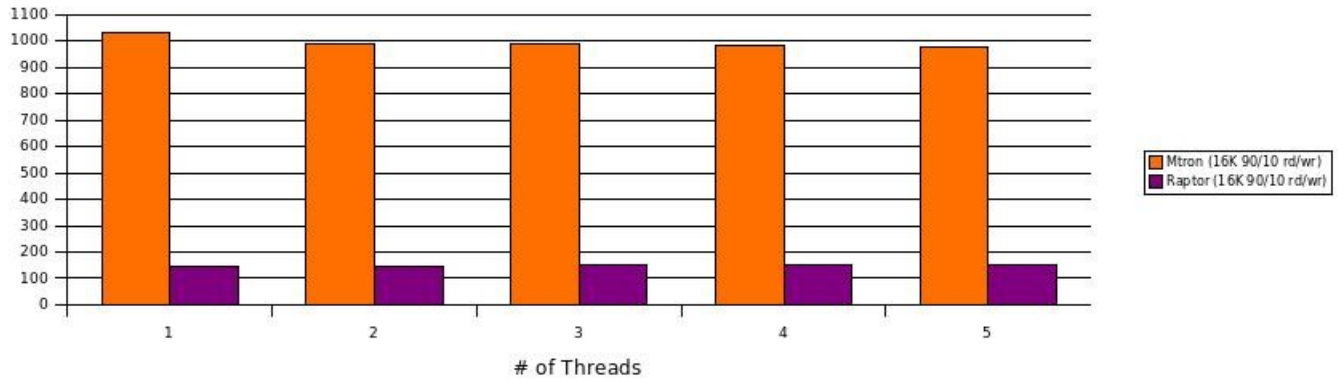


The stand alone mtron drive does beat out the raptor, only by a small margin however. Definitely not worth the price premium in this type of environment. But as we saw in sysbench, as the number of writes diminishes the # of supportable IOPS will increase. So lets look at the 80/20 & 90/10 splits:

Orion IOPS 80% reads 20% Writes



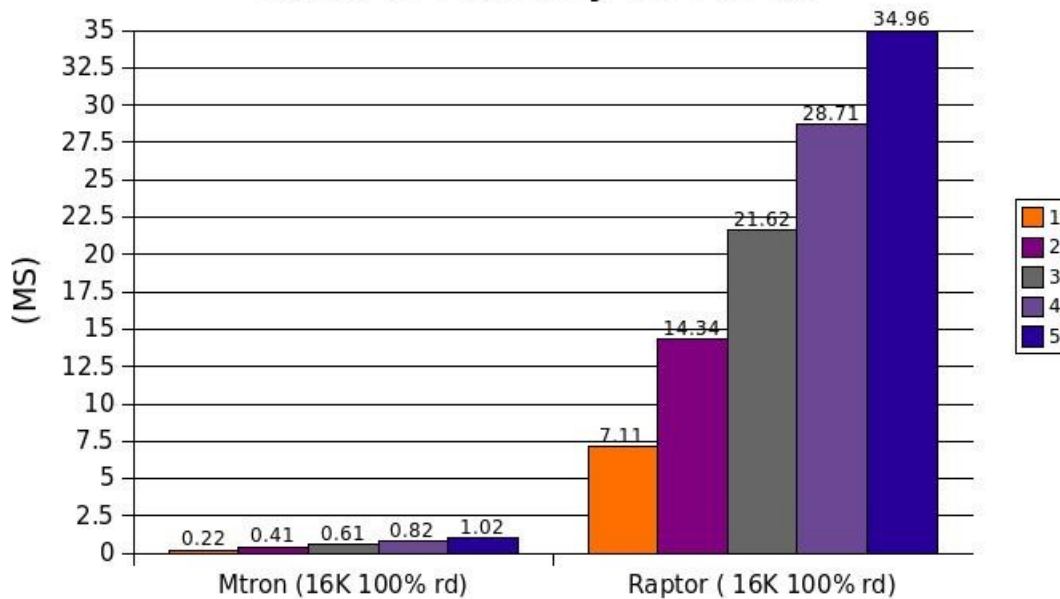
Orion IOPS 90% Reads 10% Writes



In a 90/10 type of environment the performance is about 6.5x improvement, while the 80/20 is only 3X. When your IO bound and can only fit 4-6 disks internally a few thousand dollars to see a 3-7x improvement may be very attractive to certain MySQL users. At that point your alternative is either a bigger box or an external San both of which will dwarf the cost of the 2 or 4 SSD drives. Another time I will but a controller with cache in my server to see if I can boost these numbers further. Back to the numbers... Lets take a look at latency.

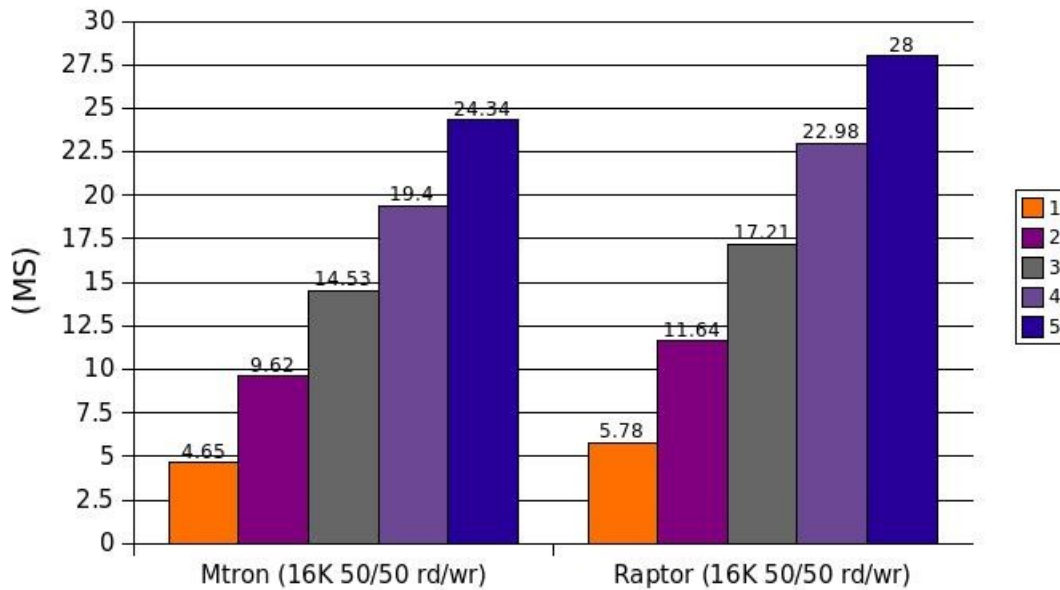
In a pure read environment:

Orion IO Latency 100% RD



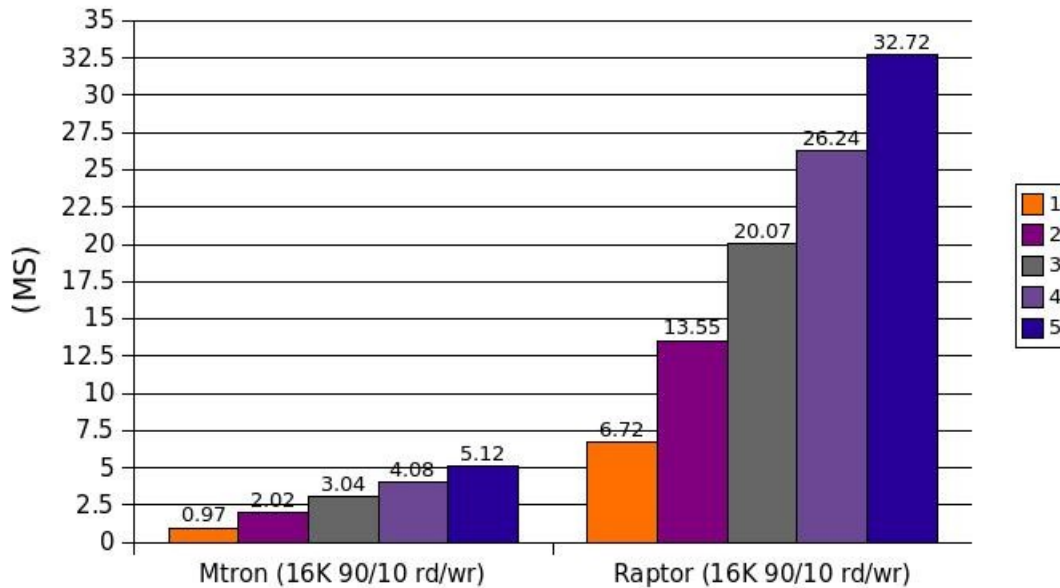
At 5 threads the latency on the Mtron drive is only 1ms vs 34.9ms on the raptor. What is not shown is that at 10 threads the Mtron drive is 2ms vs 63ms for the Raptor. In the 50-50 test the numbers are very similar:

Orion IO Latency (50-50 RD/WR)



24ms on the Mtron or 28ms on the Raptor drive is an eternity. But as I mentioned for a 50-50 environment a naked mtron or two probably is not the best solution. So what about a 90/10? Glad you asked.

Orion IO Latency (90-10 RD/WR)

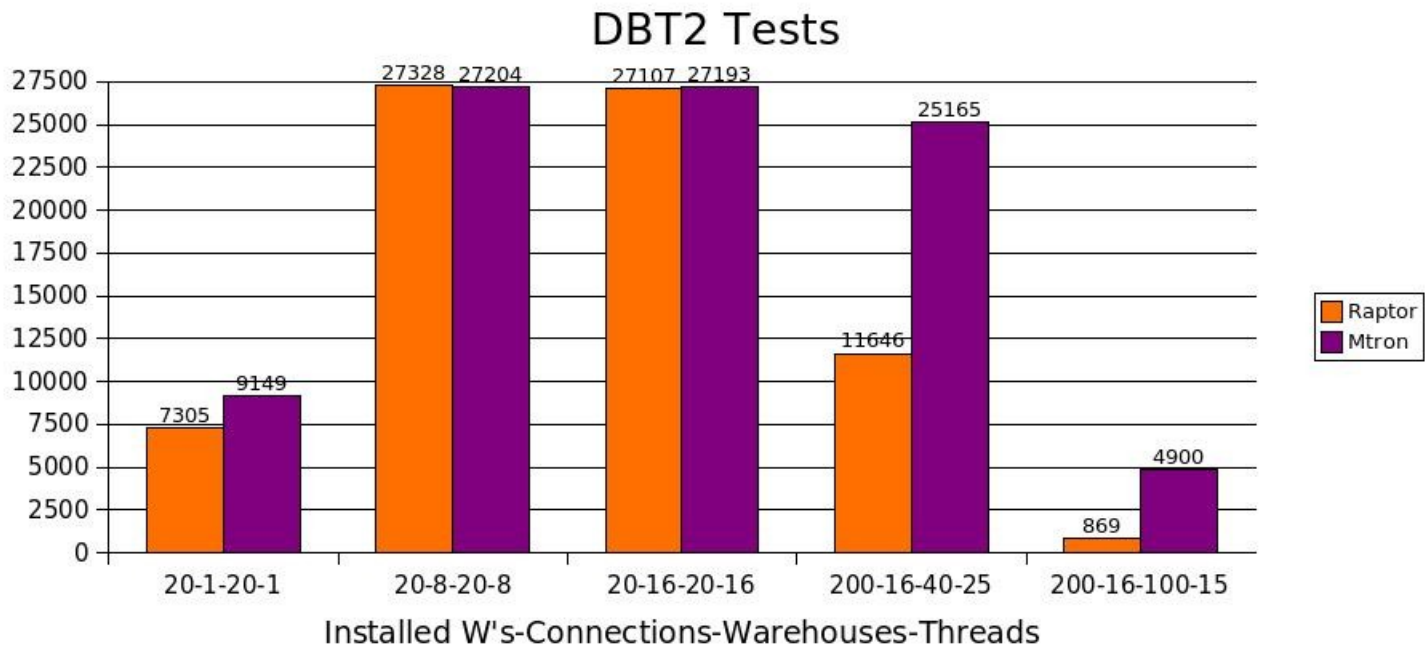


For my final trick of the day... DBT2 test results. The tests were done with the innodb storage engine. I used the following settings:

```
mysql> show variables like 'innodb%';
```

```
+-----+-----+
| Variable_name | Value |
+-----+-----+
```

```
| innodb_additional_mem_pool_size | 20971520 |
| innodb_autoextend_increment | 8 |
| innodb_buffer_pool_ave_mem_mb | 0 |
| innodb_buffer_pool_size | 2894069760 |
| innodb_checksums | ON |
| innodb_commit_concurrency | 0 |
| innodb_concurrency_tickets | 500 |
| innodb_data_file_path | ibdata1:1500M:autoextend |
| innodb_data_home_dir | /mtron/ |
| innodb_doublewrite | OFF |
| innodb_fast_shutdown | 1 |
| innodb_file_io_threads | 4 |
| innodb_file_per_table | OFF |
| innodb_flush_log_at_trx_commit | 1 |
| innodb_flush_method | 0_DIRECT |
| innodb_force_recovery | 0 |
| innodb_lock_wait_timeout | 50 |
| innodb_locks_unsafe_for_binlog | OFF |
| innodb_log_arch_dir | /data01/ |
| innodb_log_archive | OFF |
| innodb_log_buffer_size | 16777216 |
| innodb_log_file_size | 681574400 |
| innodb_log_files_in_group | 2 |
| innodb_log_group_home_dir | /data01/ |
| innodb_max_dirty_pages_pct | 90 |
| innodb_max_purge_lag | 0 |
| innodb_mirrored_log_groups | 1 |
| innodb_open_files | 300 |
| innodb_rollback_on_timeout | OFF |
| innodb_support_xa | OFF |
| innodb_sync_spin_loops | 20 |
| innodb_table_locks | ON |
| innodb_thread_concurrency | 1000 |
| innodb_thread_sleep_delay | 10000 |
+-----+-----+-----+-----+-----+
```



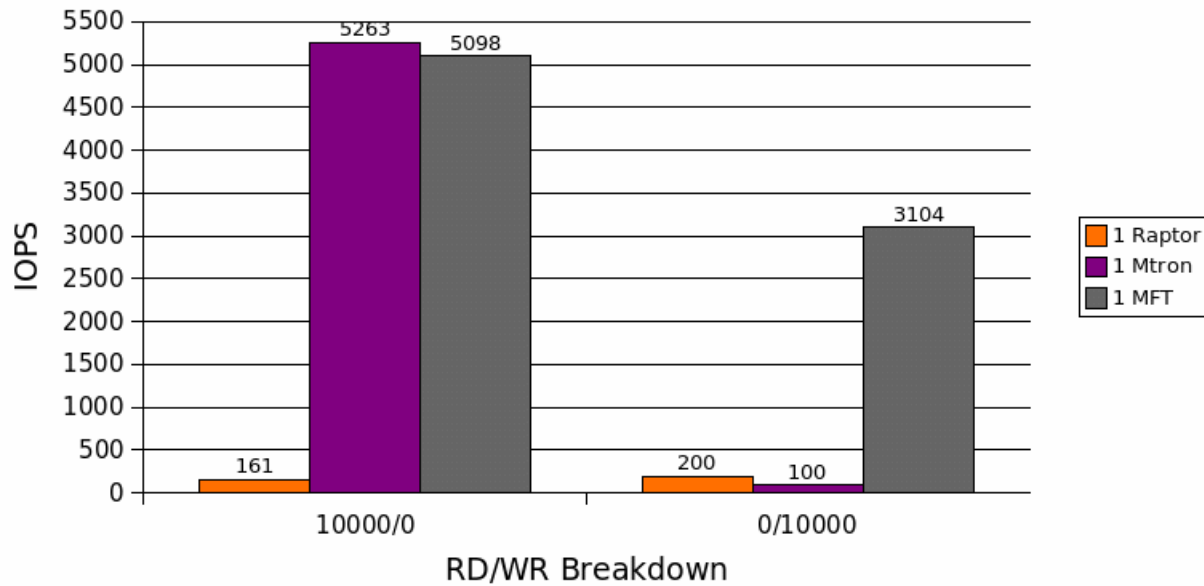
The first 3 tests can hardly be counted, they size of a 20 warehouse database is smaller then the innodb buffer pool. Effectively all this is testing is cpu and memory access. The last two results are a little more telling and accurate. Adding more warehouses and more active threads hits both disks hard, but the SSD drive is able to maintain a much higher transaction count then its platter based brother. This all makes perfect sense. The more active threads requesting data over a wider range of data, the more a traditional disk arm has to move. The more the disk arm moves, the slower the access time.

Next I am going to add in MFT technology from Easy Computing Company (<http://www.easyco.com>).

MFT Sysbench tests:

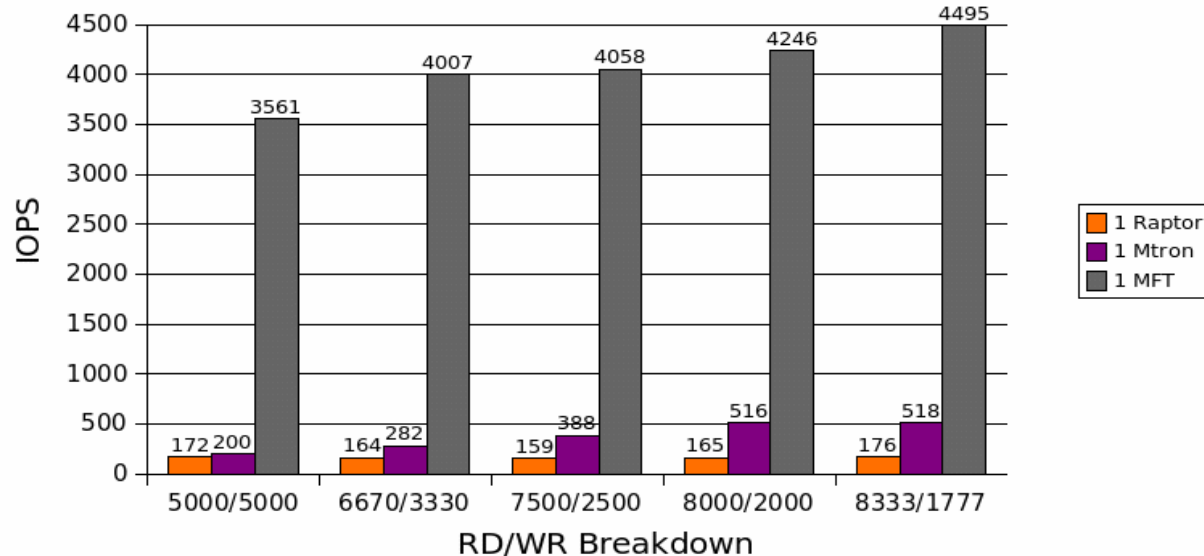
Lets look at the same numbers from sysbench with the MFT enabled drive thrown in in:

Sysbench Random RD/WR



In the generic sysbench tests the MFT enabled Mtron drive was 30x faster when performing the random write tests. What this adds up to in a mixed load test is just scary:

Sysbench Random RD/WR



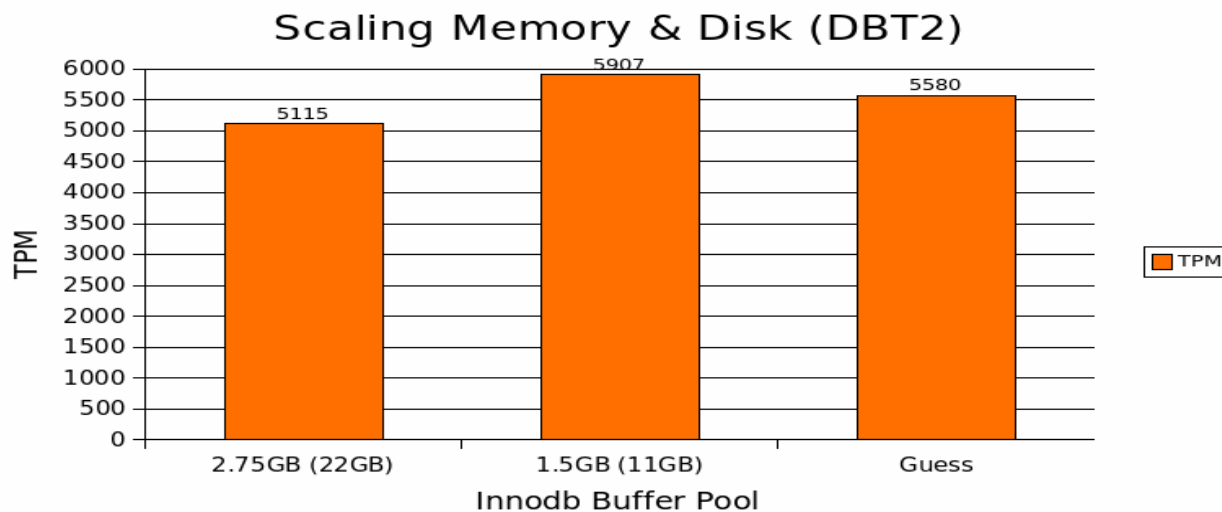
In the synthetic benchmarks the MFT/Mtron combo just blows away the competition. The Mtron drive alone is just put to shame. The best test for the standalone Mtron was a test that had 83% reads and 17% writes it was able to sustain an average of 518 IOPS. In the same test with MFT running on the same drive produced 4,495 IOPS! That ends up being almost a 9X increase.

DBT2 Tests

Lets look at the real world tests. Over the past several weeks I have gotten widely varied results from DBT2 on very similar tests. One of the puzzling items I discovered was that after performing a test and deleting the datafiles, the following test seemed to take longer to setup using MFT. For instance my initial script for testing bare Mtron drives had a 2 minute wait between tests and another 2 minute wait between the startup of MySQL and the starting of the load for dbt2. This wait was sufficient to allow all the datafiles to build after I removed them from the previous test (NOTE: I used safe_mysql, I should have used the init.d script). When I started the MFT tests, the first test was successful, but the next failed on the load. The build process for the innodb files increased dramatically, and as a result the load failed due to MySQL not being available yet. I increased the time between runs and this was fixed. I believe this to be due to how the MFT software handles writes.

Free Space:

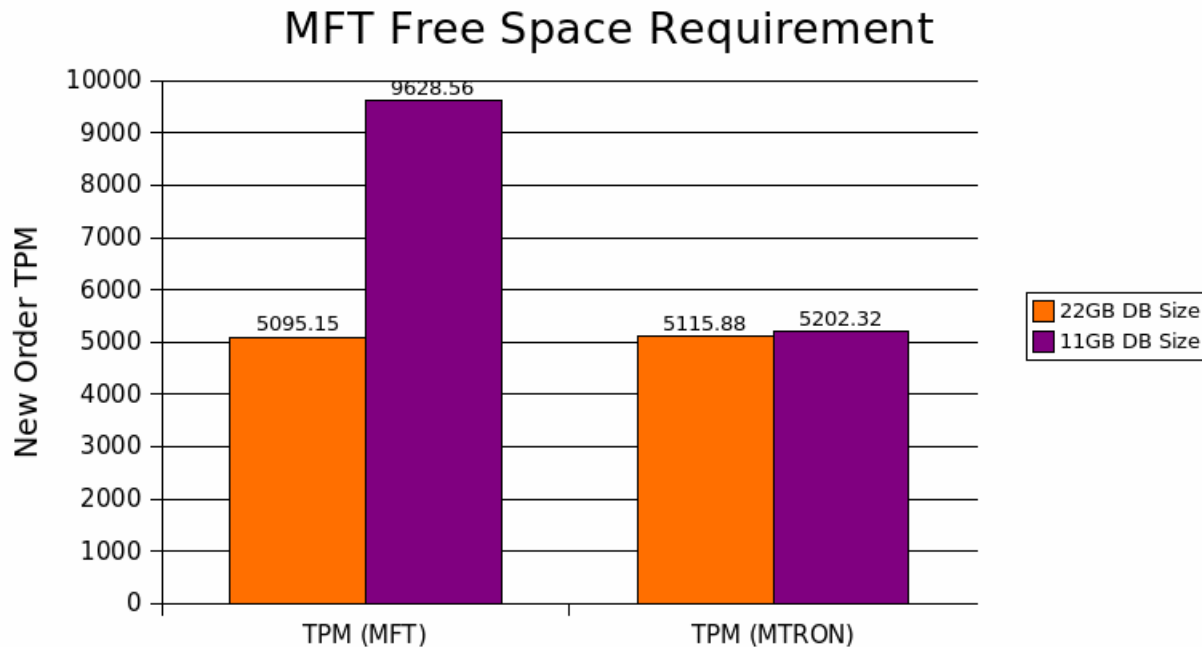
According to the folks over at EasyCo MFT relies on free space in order help speed up the random write process. Basically from what I gather it writes to the free blocks then comes back later on to clear up the old blocks. During my testing I was able to verify this. I saw an immediate improvement in relative performance when i reduced the space on the drives. In order to still flex the drive i started reducing the memory footprint on MySQL to closely match the reduction in size. If you have not done this before it works great. Basically lets say you have a 20GB data and 3GB allocated to the Innodb Buffer pool, by reducing the size from 20GB to 10GB and the memory from 3 to 1.5GB you should get similar performance #'s. Take a look here:



This is a graph of DBT2 Transactions per minute. I was running all of my tests against a 22GB database (~80% full on MFT, ~73% on bare Mtron) with 2.75GB allocated to the innodb buffer pool. Because I

was confined to a 32GB drive I reduced the space in half, and reduced the memory footprint by a little more than half. If I had to guess how many TPM I should get I would have said about 5600, instead the reduced size netted me 5900 ... about a 6% difference in what I expected. A small enough difference I decided to concentrate on using 100 warehouses (11GB) instead of the 200 warehouses (22GB).

Since I already mentioned how free space is vital in MFT performance lets look at tests comparing a 22GB db vs an 11GB db. The parameters for this test are identical with a innodb buffer size of 2.75GB and running 100 active warehouses.



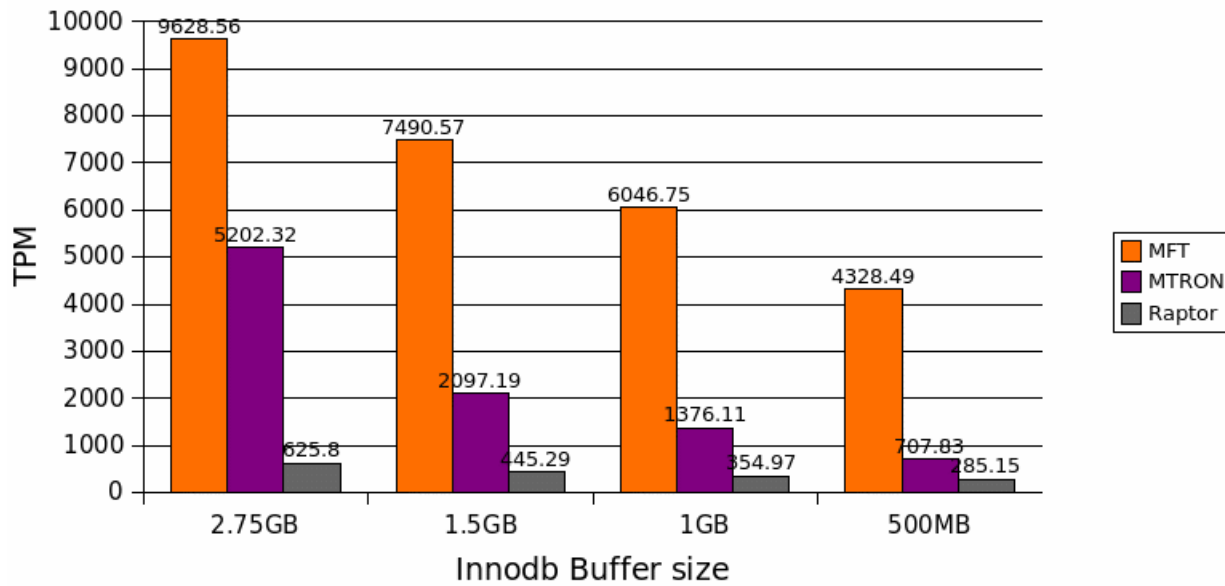
As you see above, running 100 active warehouses in the smaller database (11GB vs 22 GB) nets almost a 2X increase in performance with MFT, while on the other hand the Mtron numbers stay the same. I expected the Mtron numbers to remain constant. We are testing the same amount of active data in both tests, we are just changing the total amount of data to match the actively used size. Based on this, we have our first recommendation: if your thinking about deploying MFT with SSD plan to only fill your your drive to about 50%-60% full. If you get higher then that you risk that the performance benefit will start to shrink.

Shrinking the Buffer Pool:

Partially due to the physical storage space constraint and partially just out of curiosity, I decided to continue to shrink the buffer pools to show performance based on % of data in memory as well as try and get a better idea of actual drive performance. What I hope to determine, is how much impact is the memory is having on the physical SSD drive. Additionally we should be able to determine what sort of memory requirements you would need to duplicate the TPM #'s I was able to achieve when you introduce larger sets of data. For example using MFT I was able to achieve 9600 TPM with an 11GB DB and a 2.75 GB buffer pool. This test had a buffer pool larger enough that 25% (2.75/11) of the data

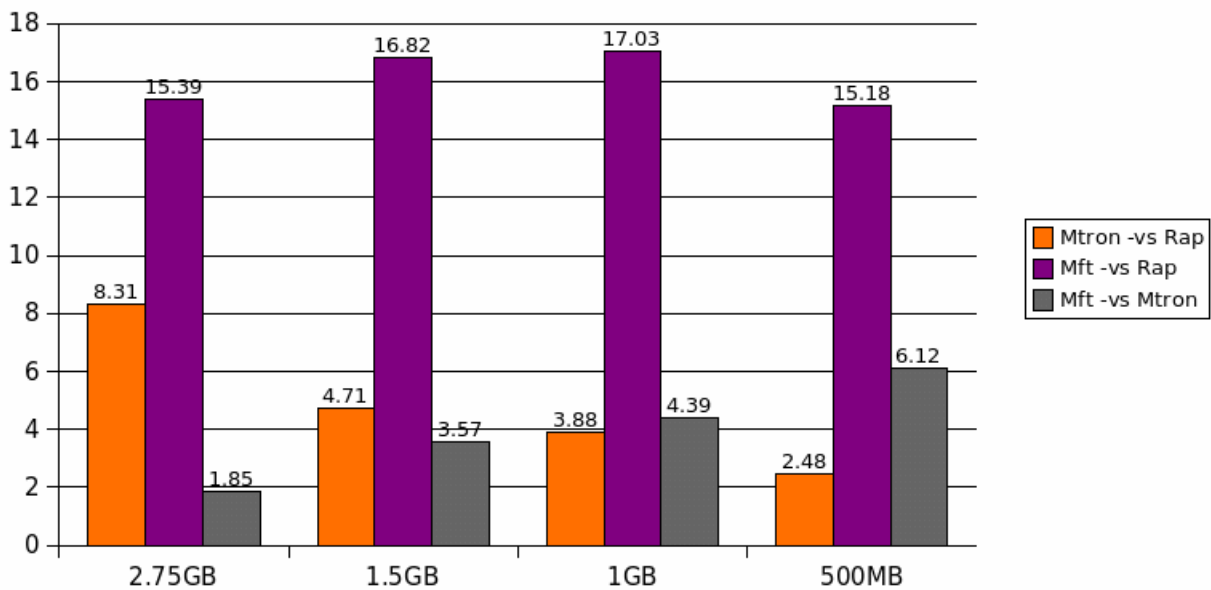
that was being used could be in memory. Using that, I can guess that if I had a 100GB database I would need to scale the buffer pool up to 25GB (25% of 100GB is 25GB) to achieve the same 9600 TPM . Lets take a look at my tests with a 2.75GB, 1.5GB, 1GB, and 500MB buffer pool:

Impact of % of data in memory



What is interesting in these numbers is that the MFT device is able maintain the high throughput even as the memory shrinks. Before I explain in more detail lets look at these in a slightly different way. Here is the X times increase over the other disks:

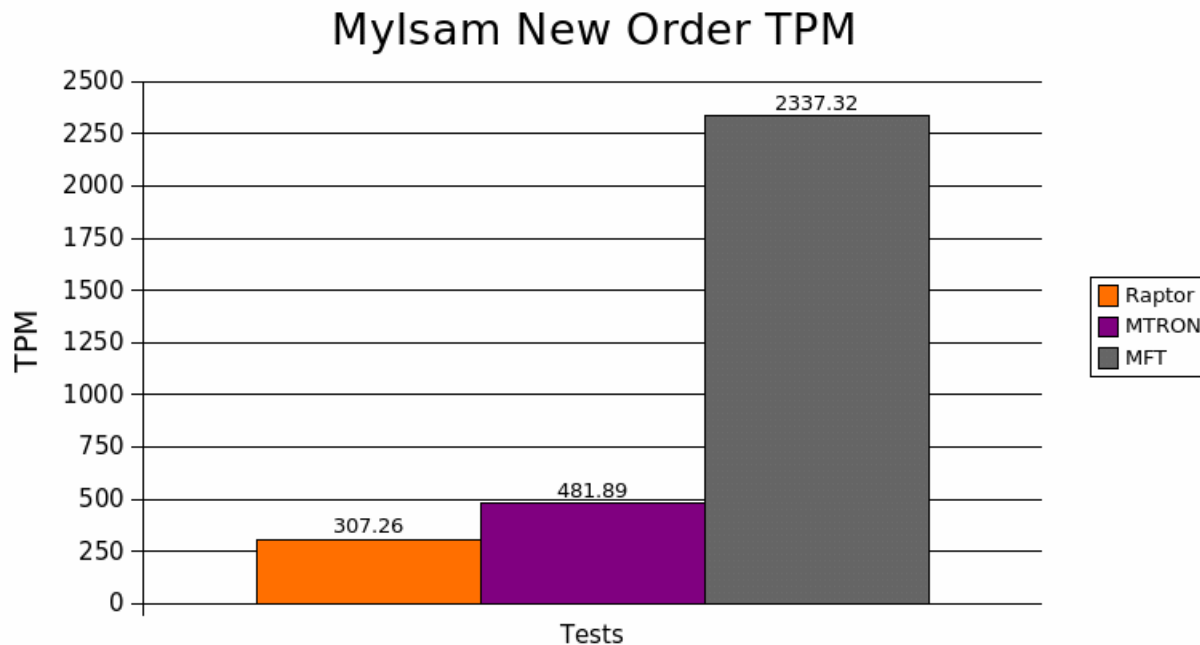
X Times Increase



Looking at when 25% (2.75GB Buffer pool) of the total data size could be in memory the Mtron drive blows the Raptor drive out of the water netting over 8X increase in transactions per minute (5202 vs 625). If you think that's good, the MFT enabled Mtron doubles that, hitting 15X over the meager Raptor, (9628 vs 625). Looking at the graphs you can definitely see a trend. As the % of memory allocated to the buffer pool compared to overall data shrinks the bare Mtrons poor write performance gets more and more pronounced. More than likely this is because there is less of a chance of having the data needed in memory as the ratio of buffer to data size decreases. Going to a 500 MB buffer pool the Mtron is just shy of 2.5X faster than the SATA Raptor. This represents a significant drop from the 8X increase the Mtron first showed. The MFT drive shows an entirely different pattern, in fact as the memory shrinks the separation between the Mtron's performance boost and the MFT's gets larger and larger, peaking at just over 6 times faster than the Mtron when the buffer pool is set to 500MB (about 4.5% of the dataset).

MyISAM Tests

I took the same dbt2 setup and ran the 100 warehouse test with a 1GB Key buffer size. Note that dbt2 locks like mad when running this test, but the results are still staggering:

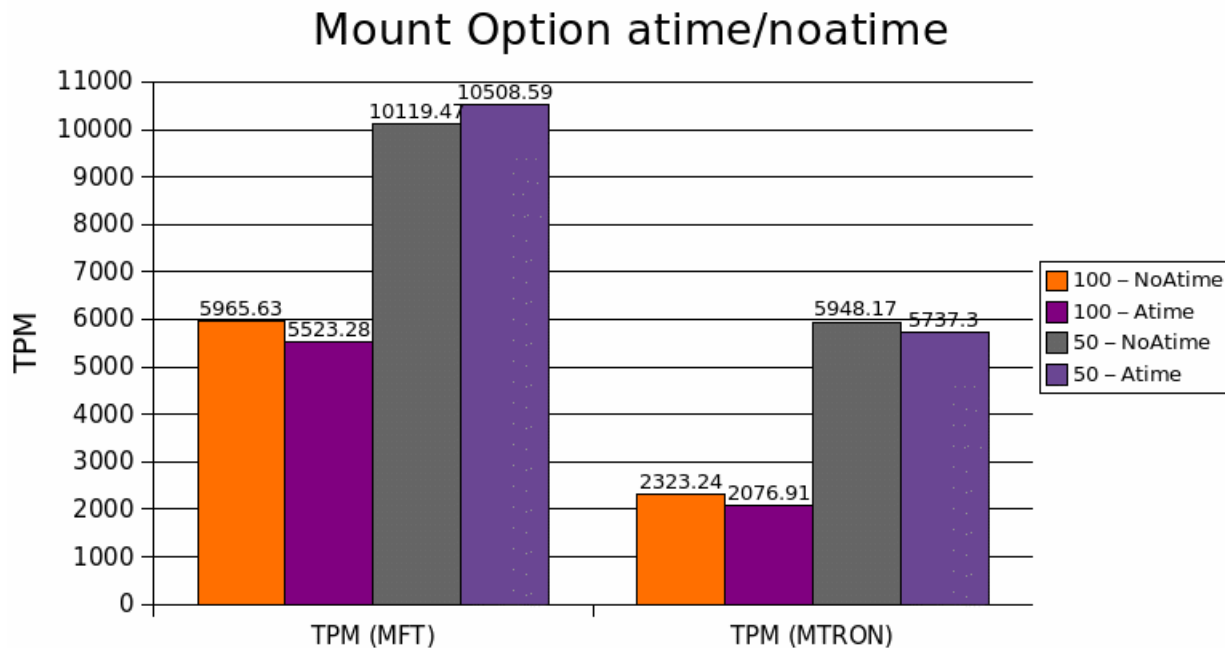


The bare Mtron drive showed a meager boost over the Raptor (about 1.5X), but the MFT enabled drive smoked both the other drives, netting a 7.6X performance boost. I guess one way to reduce locks is just to have them complete fast.

Other Tests:

Lets look at a few other options as they relate to SSD & MFT.

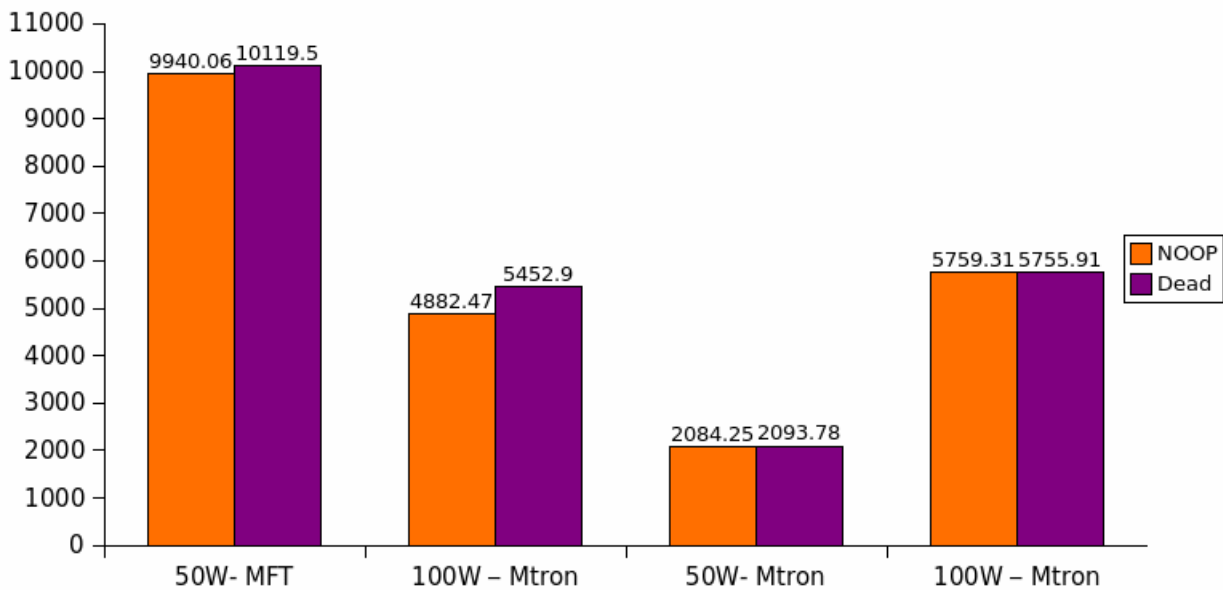
While emailing with Burton we discussed using the noatime option when mounting the drives... from a generic benchmark standpoint this looked like an awesome win. I decided to put this to the DBT2 test. So what is the TPM difference with using atime vs noatime?



Looking at the numbers the difference in new order transactions for mft is a mixed bag. Noatime on the 50 warehouse test which uses less data, saw about a 7% improvement, but the 100 warehouse test saw a decrease of about 3% (these numbers bare out over multiple tests). On the bare Mtron drive we see a small performance improvement in both tests (11% & 4%) . The interesting trend here seems to be the 7%+ drop i experienced when the active dataset was larger. Is it possible that the noatime option could help boost performance for smaller datasets but as the data gets larger there is less of an impact... hmmm that is something to try later on. For most environments it looks like noatime will give a little boost.

Next I looked at the deadline and noop schedulers specifically when using MFT & SSD.

Io Schedulers



There was not a huge boost either way when i ran my tests with the NOOP or the Deadline scheduler.

Problems:

Let me talk about a few problems I had during my testing, everything was not all high speed performance. I had 5 rather odd issues that occurred during my testing:

Problem #1:

The initial MFT driver that was sent to me was a new driver that had a rather nasty bug. When I tried to run a shell script and redirect output to the mft enabled drive (i.e. `/mft/run.sh > /mft/run.out`) after a certain amount of io the run.out file would overwrite itself. i.e. the first several lines in the file would be garbage, then what should have been the 1000th line would be the first legible thing. The folks at EasyCo sent me an older release which fixed the issue. They were right on the ball with this and very helpful in tracking down the issue.

Problem #2:

A kernel panic happened in my system. The kernel panic started when writing a lot of data to the MFT drive. At a certain point (after about 2GB in) everything would stop. Then the symptoms spread to other drives. I tried the bare mtron in ubuntu and capture the following kernel info, which is below. In

fairness after dismatling my machine, reseating the memory/controller cards ... the panic has not come back.

A quick google search shows similar panics with various kernels. But this “crash” occurred in centos & ubuntu and with several different kernels. It started 1 day with just the MFT drive, two days later the other Mtron without MFT, then about 3 days after that my internal Raptor drives. during that time frame I tried repeatedly to fix this by switching controllers, cables, etc. A simple dd would break on the MFT, while the mtron would work. Then dd broke on the mtron ... then dd broke on the Raptor. The timing was all really really strange. I can chalk this up to some strange hardware problem, and not MFT or SSD. Especially since MFT was not active in Ubuntu.

Problem #3:

MFT drive went wonky in the middle of a one (out of 280) DBT2 test. Half way through the drive went read only. Below are the errors that were raised. After fsck + a remount all was fine.

From the message log:

```
Feb 21 03:09:54 localhost kernel: EXT3-fs error (device dm-3): ext3_free_blocks:
Freeing blocks not in datazone - block = 977551942, count = 1
Feb 21 03:09:54 localhost kernel: Aborting journal on device dm-3.
Feb 21 03:09:54 localhost kernel: EXT3-fs error (device dm-3) in ext3_free_blocks_sb:
Journal has aborted
Feb 21 03:09:54 localhost kernel: EXT3-fs error (device dm-3): ext3_free_blocks:
Freeing blocks not in datazone - block = 4219567428, count = 1
Feb 21 03:09:54 localhost kernel: EXT3-fs error (device dm-3): ext3_free_blocks:
Freeing blocks not in datazone - block = 2356014811, count = 1
Feb 21 03:09:54 localhost kernel: EXT3-fs error (device dm-3): ext3_free_blocks:
Freeing blocks not in datazone - block = 1718354903, count = 1
Feb 21 03:09:54 localhost kernel: EXT3-fs error (device dm-3): ext3_free_blocks:
Freeing blocks not in datazone - block = 2452100088, count = 1
Feb 21 03:09:54 localhost kernel: EXT3-fs error (device dm-3): ext3_free_blocks:
Freeing blocks not in datazone - block = 1807742048, count = 1
Feb 21 03:09:54 localhost kernel: EXT3-fs error (device dm-3) in ext3_free_blocks_sb:
Journal has aborted
Feb 21 03:09:54 localhost kernel: EXT3-fs error (device dm-3) in ext3_free_blocks_sb:
Journal has aborted
Feb 21 03:09:54 localhost kernel: EXT3-fs error (device dm-3): ext3_free_blocks:
Freeing blocks not in datazone - block = 540315397, count = 1
Feb 21 03:09:54 localhost kernel: EXT3-fs error (device dm-3): ext3_free_blocks:
Freeing blocks not in datazone - block = 537237122, count = 300
Feb 21 03:09:54 localhost kernel: EXT3-fs error (device dm-3): ext3_free_blocks:
Freeing blocks not in datazone - block = 537237423, count = 711
Feb 21 03:09:54 localhost kernel: EXT3-fs error (device dm-3) in
ext3_reserve_inode_write: Journal has aborted
```

Problem #4:

After a reboot the MFT Drive could not mount, fsck failed on ext3. Several blocks had to be fixed.

Problem #5:

In the middle of another test run, dbt2 started throwing errors. The MySQL Error log showed disk corruption. The test was a 100 warehouse dbt2 test, with separate innodb datafiles for every table. I had run this test 4 or 5 times prior without error:

```
080227 9:32:46 InnoDB: Started; log sequence number 2 3310925664
080227 9:32:46 [Note] /usr/sbin/mysqld: ready for connections.
Version: '5.0.50spla-enterprise-gpl-log' socket: '/var/lib/mysql/mysql.sock' port: 3306
MySQL Enterprise Server (GPL)
InnoDB: Error: page n:o stored in the page read in is 1150961728, should be 395846!
InnoDB: Database page corruption on disk or a failed
InnoDB: file read of page 395846.
InnoDB: You may have to recover from a backup.
```

I talked to the folks over at EasyCo (makers of MFT) on the last 3 issues. There are some issues with the version of MFT I was using (remember I had to use an older version due to problem #1). These may be fixed in the new release that they are working on now. They said a maintenance release should be ready in a few weeks. Please note the above issues are not necessarily MFT related. The last 3 did occur on the MFT device however.

Final Thoughts and Conclusions:

While this generation of SSD disk shows a lot of promise there are some limitations to the hardware. If deployed in the proper environment it has the potential to significantly improve the performance of applications. Make sure you look at your applications characteristics, how much memory your have, etc before taking the plunge. But with the relatively low cost of the technology, you could net 10X+ performance increase on your database servers for under \$2000.

The software the MFT is working on holds a great deal of promise. Its list price adds a small premium to the overall price of the regular SSD drives, but the pay off in terms of performance will make it worth investigating and potentially owning. As with all newer technologies there are some issues, so testing the hardware/software with your workload & data is key. EasyCo has been very easy to deal with and seem eager to address the issues that I found.

In the future SSD and other vendors will streamline SSD performance, and its about time. Disk technologies have been slow to produce the ground breaking changes that have come in CPU and Video Card technologies.