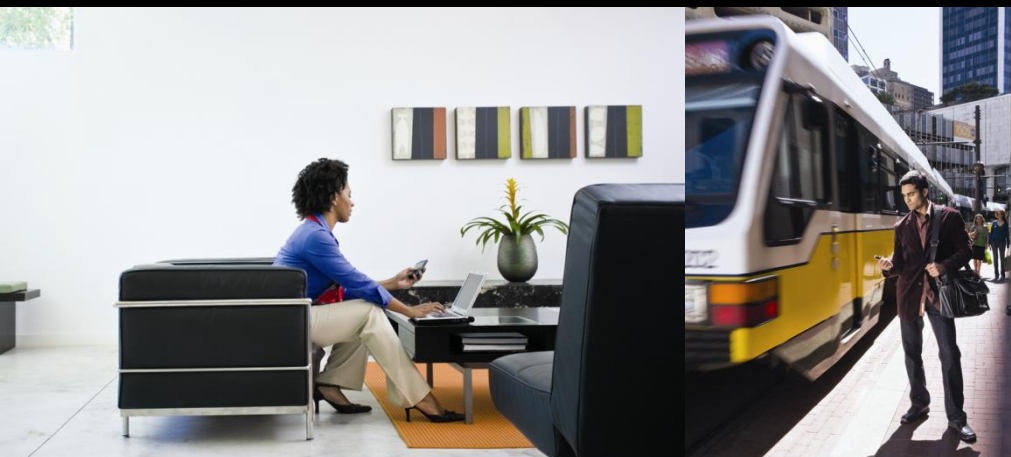




Sybase Adaptive Server Enterprise (ASE) 15.5 Performance Evaluation using Fusion-io Solid State Memory

[TECHNICAL WHITEPAPER]



Contents

- Executive Summary 1
- 1. Introduction..... 2
- 2. System Specifications 3
- 3. Fusion-io Setup and Configurations 4
- 4. The OLTP benchmark 7
 - 4.1 Operational scalability performance 8
 - 4.2 OLTP performance 10
- 5. ASE Configuration and Tunings 16
- 6. Conclusion 17
- 7. Appendix..... 18

Principal Author

Thomas Tanaka thomas.tanaka@sybase.com
Server Performance and Development Group

Revision History

Version 1.0 - January 2010

Executive Summary

Solid State Disk (SSD) is a technology that utilizes flash based solid-state memory to store data. SSD completely eliminates the use of mechanical components (e.g. disk spindle) of Hard Disk Drive (HDD), thus eliminating the risk of mechanical failure. The key benefit of SSD over HDD is a significant reduction in I/O latency for both read and write, while delivering higher IOPS, I/O bandwidth and maintaining low power consumption. SSD is increasingly being utilized in today's mission-critical and high performance enterprise systems.

This whitepaper evaluates Sybase Adaptive Server Enterprise (ASE) 15.5 performance with the inclusion of Fusion-io SSD as a choice for database storage devices. By using OLTP benchmark evaluation, we observe the following¹:

- When SSD is used for data and log, the operational scalability performance increases 7X.
- SSD speeds up database backup and recovery operation 5X.
- OLTP performance increases 1.65 – 1.81X with SSD storage.
- With an increase in the database size, the use of SSD for both data and log shows significant (13X) improvement in throughput.

¹ Mileage may vary.

1. Introduction

The SSD² that we have used in this evaluation is from Fusion-io (fio)². Fusion-io offers a different and unique implementation of the SSD technology, which utilizes the PCI Express bus interface connection. Fusion-io uses the host CPU for memory mapping, circumventing inefficient storage protocols like SAS and SATA. Fusion-io ioMemory uses Direct Memory Access, is on the system bus, and is silicon-based, making it perform more like DRAM than HDD. This allows Fusion-io SSD to deliver a much higher throughput of read and write bandwidth, IOPS and lower read/write latency compared to conventional SSD drives available in the market today.

Our evaluation of the SSD as a database device for Adaptive Server Enterprise (ASE)² uses an internal OLTP benchmark which is designed to completely simulate the typical order entry environment. This enables us to identify clearly the area in which SSD usage will or will not help increase performance. This evaluation is completed on Linux platform a similar result is expected on other platforms.

² From here onwards, Adaptive Server Enterprise will be referred to as ASE and Fusion-io SSD device will be referred to as *fio* or *SSD* interchangeably.

2. System Specifications

The table shown below describes the system specifications used.

Hardware	Dell PowerVault MD3000
CPU	24 cores - Intel(R) Xeon(R) E7450 @ 2.40GHz
Memory	64GB
OS	Red Hat Enterprise Linux (RHEL) Server release 5.2 (Tikanga) Linux Kernel 2.6.18-92.el5 #1 SMP x86_64
HDD	13 x 450GB 15K RPM Serial-Attach SCSI 3Gbps 3.5-in
SSD	Fusion-io ioDrive 320GB - MLC Fusion-io ioDrive 160GB - SLC
Software	Fusion-io driver version: 1.2.6.5 Adaptive Server Enterprise (ASE) 15.5

3. Fusion-io Setup and Configurations

Installation packages for the fio device can be obtained from <http://support.fusionio.com/>. The driver is available in various Linux distributions and Windows. The necessary installation packages are *iodrive-driver* and *iodrive-util*. The user guide is available at the same location under the *documentation* section. Run a proper package manager for installation, in the case of RHEL, it is `rpm`.

	Location
Driver	/lib/modules/<kernel-version>/fusionio/
Utilities	/usr/bin

When running all of the command relating to fio, root privilege is required.

Install *iodrive-driver* and *iodrive-util* using `rpm` package manager:

```
# rpm -i iodrive-driver-1.2.6.5-1.0_2.6.18_92.el5.x86_64.rpm
# rpm -i iodrive-util-1.2.6.8-1.0.x86_64.rpm
```

After installation is successful, run the following command to load the driver:

```
# modprobe fio-driver
```

By default, the driver is loaded automatically during system boot; it is available as `/dev/fiox`, where `x` is a letter (i.e., a, b, c, etc.) labeled with the number of drivers installed in the system. The driver is then available to be used similarly to a block device.

To check the status of fio installed in the system:

```
# fio-status
```

The following is sample output:

```
Found 2 ioDrives in this system
Fusion-io driver version: 1.2.6.5
```

```
Adapter: ioDrive
```

```
    Fusion-io ioDrive 320GB, Product Number:FS1-002-321-CS SN:18015
```

```
    PCIE Power limit threshold: 24.75W
```

```
    Connected ioDimm modules:
```

```
    fct0: Fusion-io ioDrive 320GB, Product Number:FS1-002-321-CS SN:06291
```

```
fct0    Attached as 'fioa' (block device)
        Fusion-io ioDrive 320GB, Product Number:FS1-002-321-CS SN:06291
        Alt PN:FS1-SS2-321-CS
        Firmware v36867
        322.12 GBytes block device size, 396 GBytes physical device size
        Internal temperature: avg 51.7 degC, max 54.1 degC
        Media status: Healthy; Reserves: 100.00%, warn at 10%
```

Adapter: ioDrive

```
Fusion-io ioDrive 160GB, Product Number:FS1-002-161-ES SN:15596
PCIE Power limit threshold: 24.75W
Connected ioDimm modules:
fct1: Fusion-io ioDrive 160GB, Product Number:FS1-002-161-ES SN:10858
```

```
fct1    Attached as 'fiob' (block device)
        Fusion-io ioDrive 160GB, Product Number:FS1-002-161-ES SN:10858
        Alt PN:FS1-SS2-161-ES
        Firmware v36867
        160.92 GBytes block device size, 198 GBytes physical device size
        Internal temperature: avg 47.2 degC, max 51.2 degC
        Media status: Healthy; Reserves: 100.00%, warn at 10%
```

To perform a disk format, use the `fio-format` utility (Note that all data will be lost.). For example, to format the `fioa` block device:

```
# fio-detach /dev/fct0
# fio-format /dev/fct0
# fio-attach /dev/fct0
```

Use the `fdisk` command for partitioning. For example, to partition `/dev/fioa` in a Linux system:

```
# fdisk /dev/fioa
```

The `fio` device can also be used as a raw device. For example, to create a raw device for `/dev/fioa1`, and assign it to `/dev/raw/raw1`:

```
# raw /dev/raw/raw1 /dev/fioa1
```

To use fio as a filesystem, please refer to the performance and tuning guide provided by Fusion-io. For the purpose of this evaluation, we have used fio as a raw device for the OLTP data and log. We used fio as the `ext3` filesystem to store the database dump.

One important point to note is that `fctx-worker` and `fiox-data-groom` are Fusion-io daemons that are run periodically to ensure the consistency of driver performance. Here is the `top` output while the fio driver is in use.

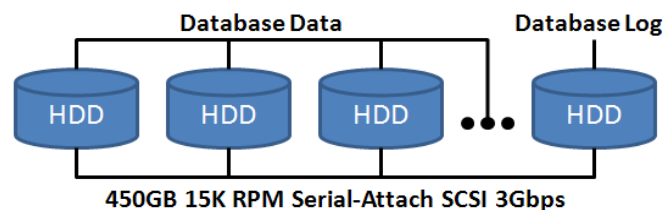
<u>PID</u>	<u>USER</u>	<u>PR</u>	<u>NI</u>	<u>VIRT</u>	<u>RES</u>	<u>SHR</u>	<u>S</u>	<u>%CPU</u>	<u>%MEM</u>	<u>TIME+</u>	<u>COMMAND</u>
8791	root	10	-5	0	0	0	S	73	0.0	87:16.48	fct1-worker
4806	root	10	-5	0	0	0	S	29	0.0	72:52.05	fct0-worker
11481	sybase	15	0	17876	2208	908	S	4	0.0	0:00.77	top
8926	root	10	-5	0	0	0	S	3	0.0	36:44.84	fioa-data-groom
8985	root	10	-5	0	0	0	S	2	0.0	0:27.94	fiob-data-groom

`fctx-worker` and `fiox-data-groom` daemons make use of idle CPU cycles to ensure that the system is always fully utilized. However, the rates at which these daemons run are changeable. Generally, under common enterprise system settings, the daemon's CPU consumption rate under the default setting does not pose significant threat to the overall system performance. Please refer to the Fusion-io performance tuning guide to change the default rate setup. In this evaluation, we have installed two Fusion-io ioDrives: ioDrive 320 GB MLC and ioDrive 160 GB SLC.

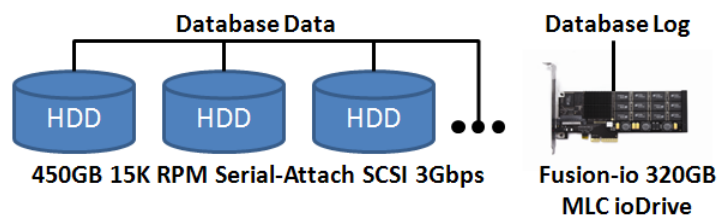
4. The OLTP benchmark

The OLTP benchmark simulates an order-entry environment that represents real life business activities in order placement scenarios. It covers operations like new order placements, entry of customer payments, checks on order status, directives to fulfill shipment for orders placed and queries on updated inventory levels. Close to half the total number of transactions are new incoming orders. Transactions include processing, concurrency control, logging and various DML and database query operations.

We used three storage layout setups for this experiment. The first setup uses only HDD, where we use a RAID 0, i.e., stripping data and segments across multiple HDDs, with one stand-alone HDD solely reserved for transactions logging. The high-level setup is shown in the diagram below.



The second setup uses SSD instead of HDD for transactions logging. This setup makes use of the higher write bandwidth and IOPS that SSD technology is able to deliver, thus enabling us to evaluate the possible performance gain. The high-level setup diagram is shown below.

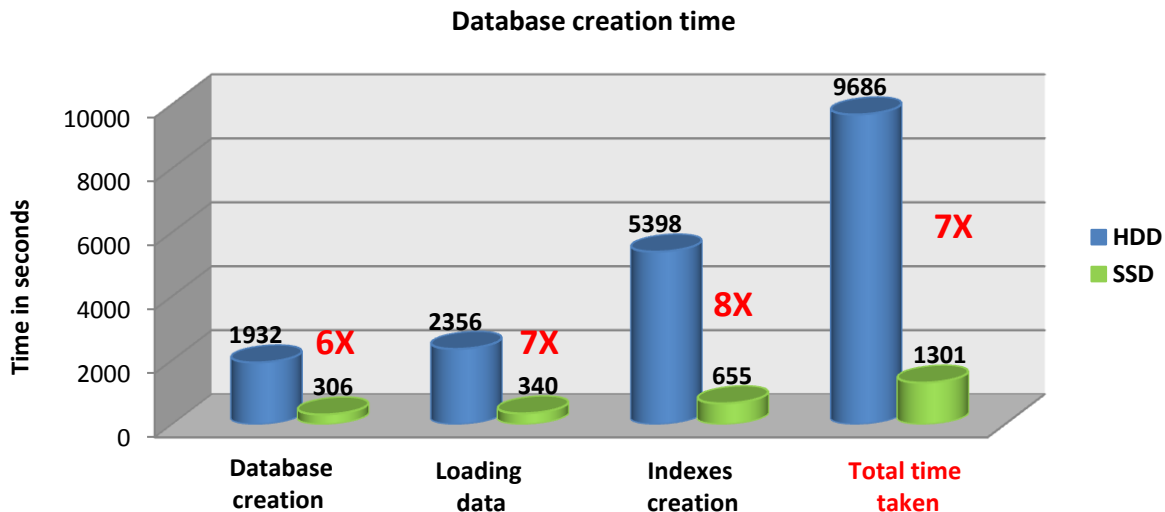


The third setup uses a single SSD drive, which has three partitions. The first partition is used for data, the second partition is for transaction logging and the third partition is used for a user filesystem to store the database dump.

4.1 Operational scalability performance

4.1.1 Database building performance

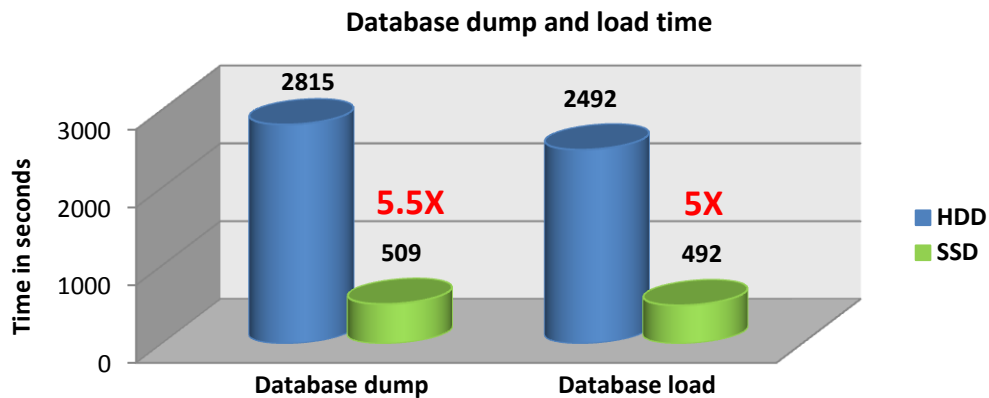
The very first step involved in the OLTP benchmark is the building of the database. This consists of three main steps: creating the database, loading data and creating indexes. For this part of the experiment, we are only using all HDD and all SSD setups. Creating devices, the database and segments is the initial step. This step involves creating and initializing devices to be used for creating the database. In the data loading step, bulk copy transfers large amount of data to the device. A gain of almost **7X** is observed with the SSD setup compared to the HDD setup. Following the load operation, indexes are created on specified columns of tables in the database. This process also involves sorting un-ordered data based on the index key, the longest part of the total loading time. The SSD completes this operation **8X faster** than the HDD setup. The chart below summarizes the duration of each of the operation (*lower is better*).



The benefit of using SSD technology is clearly visible from the chart, which shows of the time spent during the three steps. The total time in the case of SSD is only 22 minutes and 35 seconds. Compare this to HDD, which takes 161 minutes and 26 seconds (~3 hours). The overall increase in time is **7X faster** for SSD.

4.1.2 Database backup and recovery performance

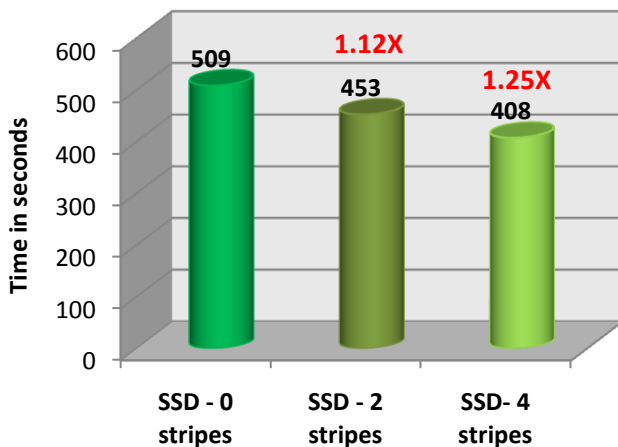
Periodic database backup is necessary to avoid unexpected events that might contribute to data loss or corruption. Dumping the database creates a database backup. The recovery process can then read from the database dump and restore the database accordingly. With increases in the database size, the backup and recovery time also increases. The impact of utilizing SSD as database device is the significant reduction of database dump and load time. The database used for this experiment is approximately 95GB with nine tables and several indexes. High sequential read and write operations are to be expected during these two stages. The database dump and load is performed when there is no other user activity present in the ASE server. The database dump and load times are shown below (*lower is better*).



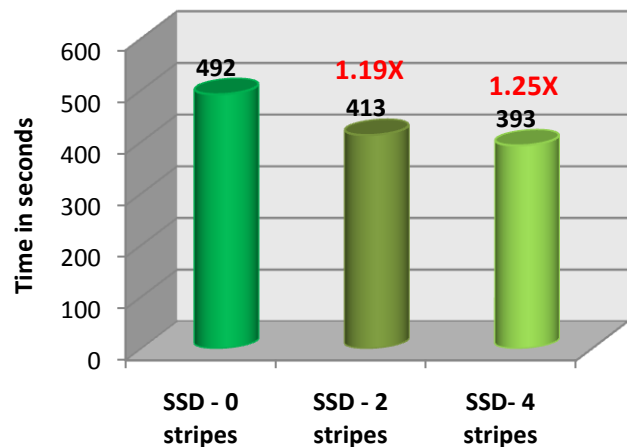
The all SSD setup performs better than the all HDD setup and is able to dump the database in 8 minutes. Similarly, during the database load, the SSD setup is **5.5X faster** in database dump time and **5X faster** in database load time when compared with the HDD setup. For the SSD setup, we mounted the third partition as the `ext3` filesystem type for our database dump storage. For the HDD setup, the database is dumped on HDD mounted with the `ext3` filesystem.

The dump and load time in the SSD setup can be further reduced up to **1.25X faster** by using multiple stripes. See the chart below, which compares all SSD setups.

Database dump on SSD with different stripes



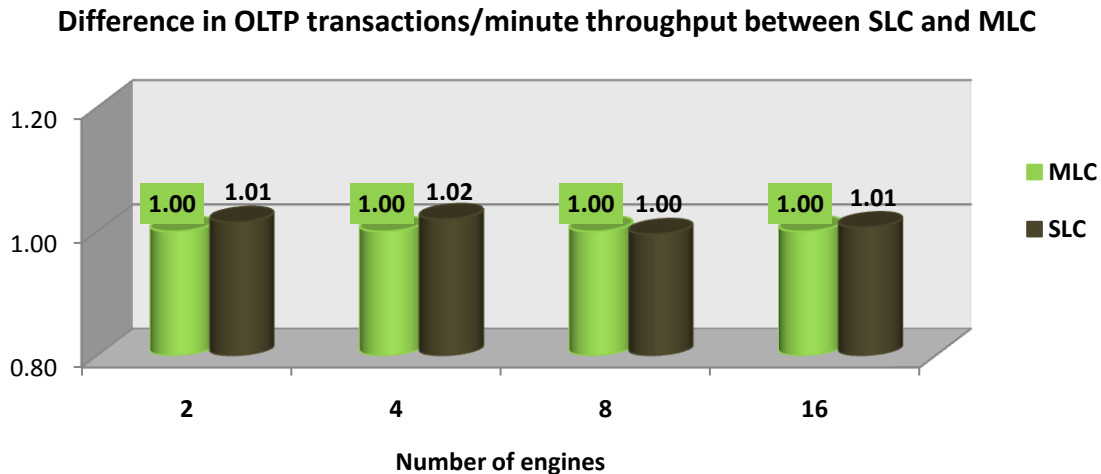
Database load on SSD with different stripes



4.2 OLTP performance

4.2.1 Performance of Multi Level Cell (MLC) versus Single Level Cell (SLC) Fusion-io ioDrive

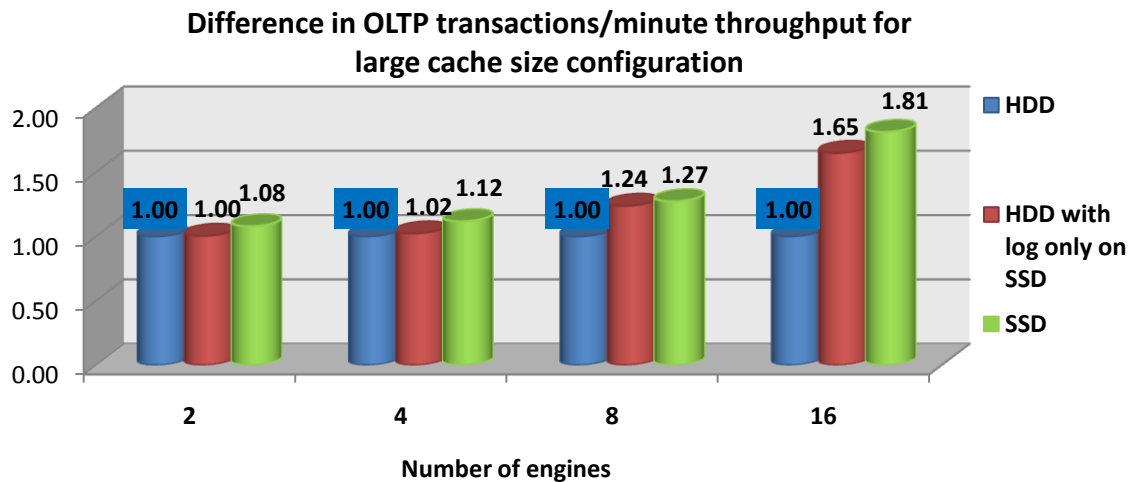
MLC stands for Multi Level Cell and SLC stands for Single Level Cell. The key technology that distinguishes the two is that, SLC is faster (in terms of read/write IOPS and bandwidth) and has a higher rate of write-erase cycles than MLC. Price-wise, MLC is cheaper than SLC. Fusion-io SLC ioDrive has ~100K read/write IOPS compares to ~70K for MLC ioDrive. The difference in the OLTP performance between SLC and MLC is shown in the chart below.



Based on the OLTP benchmark evaluation, no significant performance difference between MLC and SLC is observed (*SLC performed ~1% better than MLC as shown from the chart above*). The highest read/write IOPS of the OLTP benchmark for this particular experiment is ~25K. However, when an I/O performance measuring tool (e.g., fio-1.35) is used, we verified that SLC performed much better than MLC under random and sequential read/write IOPS and bandwidth tests. For the purpose of this evaluation, we used the Fusion-io 320 GB MLC ioDrive because it has a larger storage capacity and exhibited no performance difference when compared to the Fusion-io 160 GB SLC ioDrive.

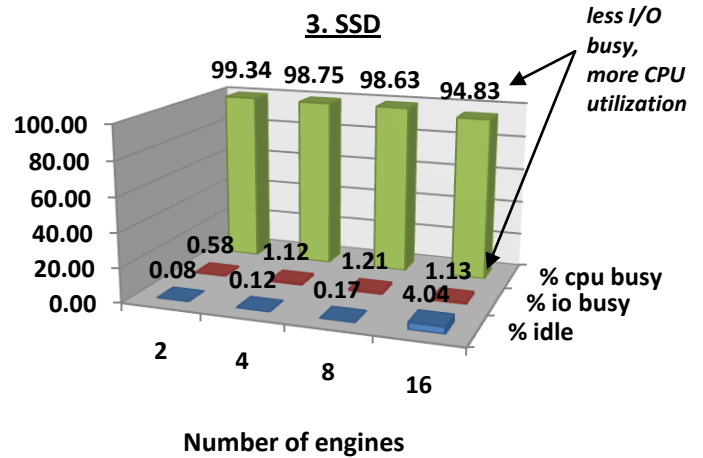
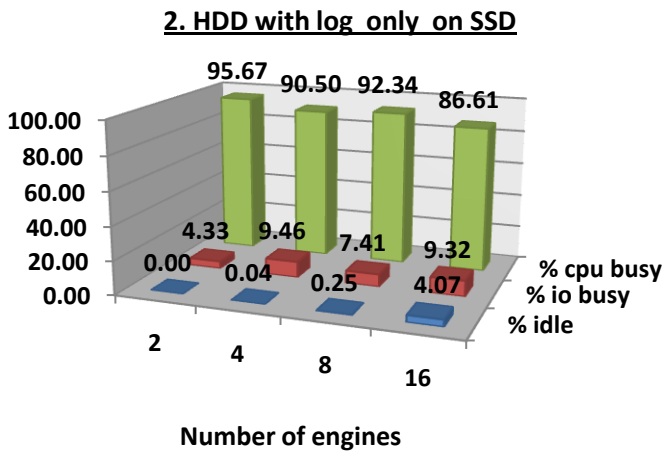
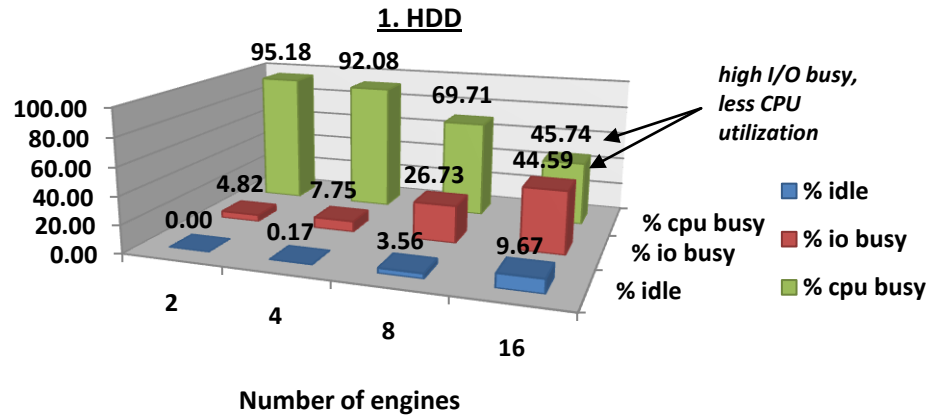
4.2.1 Performance with large caches configuration

For the benchmark run, we used a varying number of clients with a varying number of ASE online engines to evaluate the throughput by measuring transactions per minute. Engines in the ASE environment correlate to the number of CPU cores utilized by the server. By varying the number of clients and engines, our goal is to ultimately reach the state whereby the CPU cores are fully utilized (>90%) and the increase in the number of client connections does not affect performance metrics. The chart below shows the difference in throughput among the three setups (*higher is better*).



Using large size caches enables us to store more data, especially hot tables and indexes in the cache. This, uses the system physical memory which reduces I/O latency time. Creating multiple named caches for hot tables and binding them using *sp_bindcache* is necessary to drive higher throughput. During the run, the state of the server was monitored using the built-in ASE system monitoring tools (*sp_sysmon*) and collecting snapshots of performance data for various workloads that the server was executing.

sp_sysmon outputs showing % CPU and % I/O utilization for all three setups

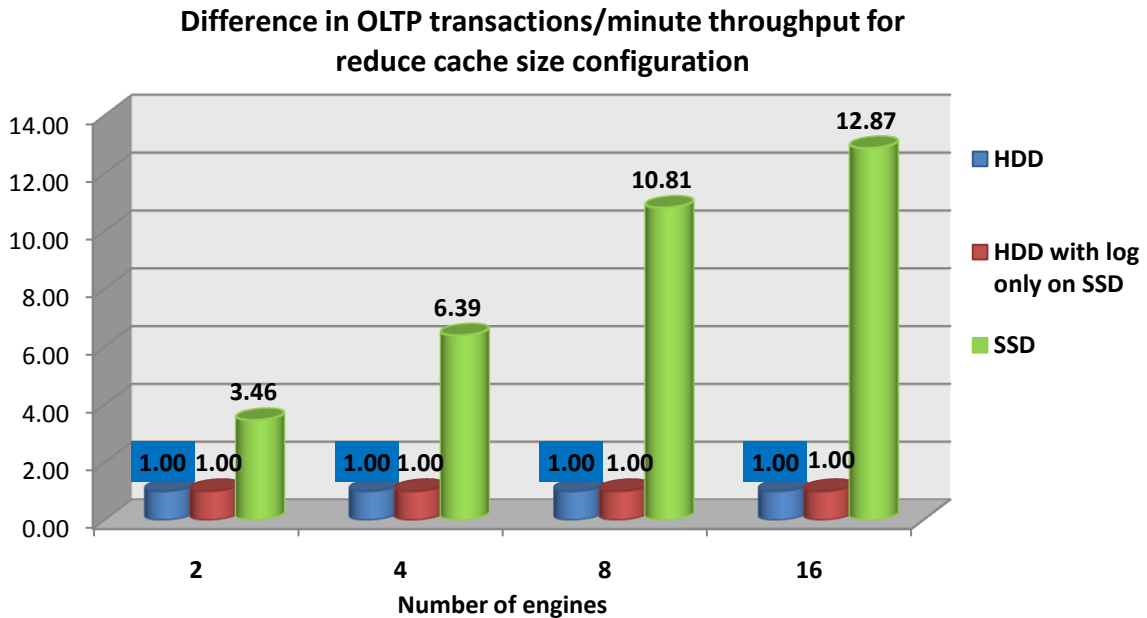


The charts above show the average CPU and I/O utilization for all three setups when ASE is running with 2 to 16 engines. The increase in the number of engines lets the server handle more transactions. As more transactions take place, the logging activities increase correspondingly; or, in other words, the increase in the number of transactions increases the write operations to the transactional log device. From the all HDD setup (1), we observe that the CPU utilization decreases as we increase the number of engines and the percentage I/O busy increases accordingly. This is expected, because the server is waiting to complete the I/O because HDD has a high latency time. By using SSD as the logging device only (2), we are able to reduce write latency time and therefore reduce the percentage of I/O busy from 45% to 9%. With this setup, we observed an **increase in the number of transactions per minute of 1.65X**. The one important point to observe from the OLTP benchmark chart is that there is ~15% performance difference between all SSD (3) and log only SSD (2) setups because the CPU is almost fully utilized. The key points from this experiment are:

- When ASE's CPU utilization is already high (>90%) significant performance throughput using SSD is not observed – refer to the results of 2 and 4 engines.
- On a system with high volume of transaction log writes, using SSD helps in increase transaction throughput due to lower I/O latency and higher IOPS in write operations. – refer to log-only on SSD setup.
- Using SSD setup for log-only **increases the transaction throughput by at least 1.65X** and using SSD setup for both data and log **increases the transaction throughput by at least 1.81X** as compared to the traditional (all HDD) setup.

4.2.2 Performance with reduced caches configuration

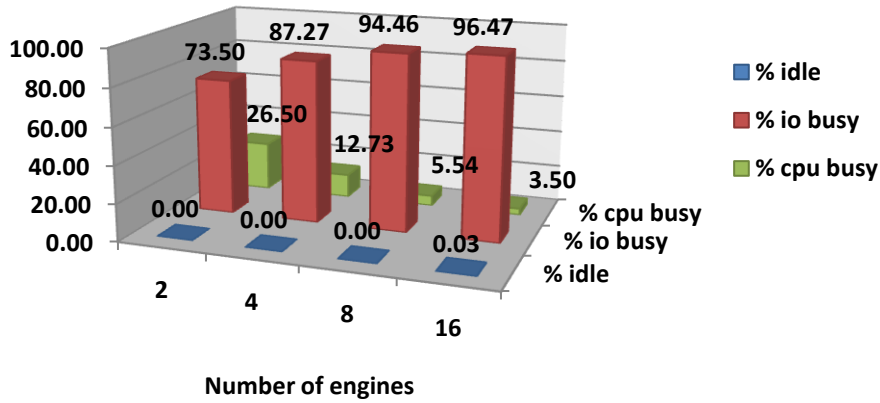
The main goal of this experiment is to simulate a database system, that is subject to a high data growth rate. With the increase in data size, cache size has to be increased correspondingly to avoid I/O penalty from the disk access. Therefore, for this particular experiment, the caches size are reduced to create the scenario. The chart below shows the difference in OLTP transactions throughput in the event of a significant increase in data size among the three setups (*higher is better*).



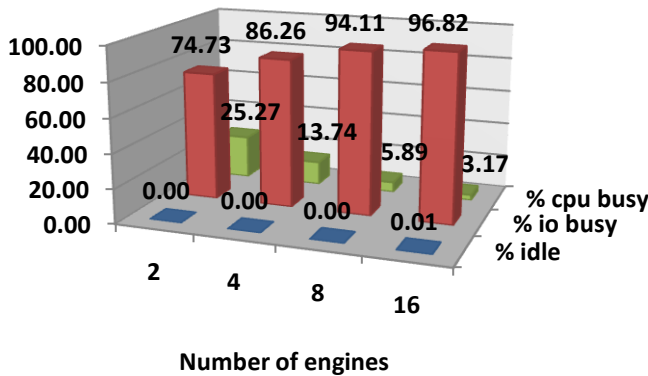
When we increased the data size by ~8X, we observed an **~13X increase in transactions per minute** in all SSD setup using 16 engines. All data and logging on SSD is able to maintain transaction throughput due to the low I/O latency time. The log-only on SSD setup performed almost equally as well as the HDD setup. This is due to the majority of hot tables and indexes stored in HDD that contribute to the high I/O latency time.

sp_sysmon outputs showing % CPU and % I/O utilization for all three setups

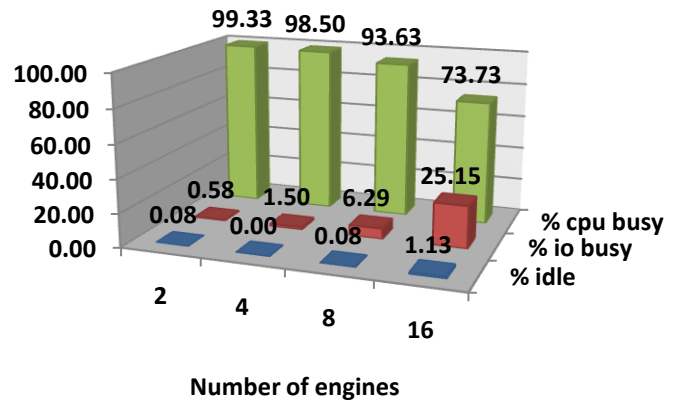
1. HDD



2. HDD with log only on SSD



3. SSD



From the sp_sysmon charts above, we notice that the all SSD setup (3) gives us the best CPU utilization and this attributes to the reduction of the I/O busy percentage with respect to the other two setups. The I/O busy percentage increases (due to the increase in the data size) as the number of engines increases in all three setups. The low I/O latency in the case of SSD significantly boosts performance. The log only on SSD setup (2) did not help much for this experiment, since hot tables and indexes reside on HDD, which has high I/O latency. The key points from this experiment are:

- In the case of the system subject to increases in data, using SSD significantly improves performance, due to the lower I/O latency that reduces the I/O bottleneck.
- For the similar system, using log only on SSD may not improve performance throughput, due to the high I/O latency time of HDD during table and index access.

5. ASE Configuration and Tunings

The followings are the general ASE configuration and tunings used in this evaluation and recommendations (*refer to appendix for a complete lists of non-default settings*).

1. Use multiple named caches with well-partitioned settings, ideally one named caches for each table in the database and for the log. The example below shows the configuration file settings of a named cache for a customer table and log only named cache.

```
[Named Cache:c_customer]
    cache size = 1000M
    local cache partition number = 32

[Named Cache:c_log]
    cache size = 500M
    cache status = log only
```

2. Use `sp_bindcache` to bind the named cache created as above. For example, `sp_bindcache, "c_customer", "db1", "customer"`. Syntax: `sp_bindcache cache_name, database_name, table_name`.
3. Use `des_bind` option to bind a table to an object descriptor for persistent storing in a metadata cache. Syntax: `dbcc_tune('des_bind', database_id, table_name)`.
4. Use `sp_sysmon` to better identify the bottleneck in the deployed database system.
5. Increase the maximum physical memory size.
6. In the system with multiple CPUs, increasing the number of ASE engines increase performance throughput. It is also crucial to bind a particular engine to a CPU by using the `taskset` command in Linux.
7. Increase the network packet size to enable larger amounts of data to be sent/received.
8. For this evaluation, we have used a 4k (4096) page size.

6. Conclusion

SSD offers low I/O latency, a superior read and write bandwidth and IOPS when compared to HDD. Higher throughput in terms of transactions per minute is observed when SSD is the choice of storage device for ASE. The metric for performance comparison however depends also on overall system price. Since the price difference in terms of \$/GB comparing SSD and HDD is still significantly high, the tradeoff between performance gain versus price ratio has to be evaluated in details to ensure overall positive gain from the SSD adoption.

With respect to the OLTP benchmark that we ran during our evaluation of SSD usage in the ASE environment we have the following conclusions:

- Using an all SSD device speeds up the operational scalability by up to 7X³ when compared to an all HDD setup.
- SSD usage for database backup and recovery reduced the operation time by up to 5X³.
- Using SSD for a log-only device in optimal ASE configuration, increases throughput by 1.65X³; including both data and logging in SSD with an optimal configuration, increases throughput up to 1.81X³.
- In a system experiencing data growth, when cache sizes are insufficient to store most hot tables and indexes, moving all data and logging to an SSD device increases performance throughput up to 13X³.
- When CPU utilization is high (>90%), using an SSD setup does not increase the performance.
- SSD gives a performance boost when I/O is determined to be the primary bottleneck.

³ Mileage may vary.

7. Appendix

ASE configuration file for non-default settings:

```
[Named Cache:c_customer]
  cache size = 3200M
  cache status = mixed cache
  local cache partition number = 32

[4K I/O Buffer Pool]
  pool size = 2200M
  wash size = 4096 K

[Named Cache:c_customer_index]
  cache size = 320M
  cache status = mixed cache
  cache status = HK ignore cache
  local cache partition number = 32

[Named Cache:c_history]
  cache size = 320M
  cache status = mixed cache
  local cache partition number = 32

[Named Cache:c_log]
  cache size = 3200M
  cache status = log only
  local cache partition number = 32

[4K I/O Buffer Pool]
  wash size = 1024 K

[16K I/O Buffer Pool]
  pool size = 100.0000m

[Named Cache:c_no_ol]
  cache size = 6400M
  cache status = mixed cache
  local cache partition number = 32

[Named Cache:c_ol_index]
  cache size = 320M
  cache status = mixed cache
  local cache partition number = 32

[4K I/O Buffer Pool]
  pool size = 300M
  wash size = 4096 K

[Named Cache:c_orders]
  cache size = 3200M
  cache status = mixed cache
  local cache partition number = 32

[Named Cache:c_stock]
  cache size = 6400M
  cache status = mixed cache
  local cache partition number = 32

[4K I/O Buffer Pool]
  wash size = 4096 K

[Named Cache:c_stock_index]
  cache size = 32M
  cache status = mixed cache
  cache status = HK ignore cache
  local cache partition number = 32
```

```

[Named Cache:c_tinyhot]
  cache size = 320M
  cache status = mixed cache
  cache status = HK ignore cache
  local cache partition number = 32

[Named Cache:default data cache]
  cache size = 500M
  cache status = default data cache
  cache status = HK ignore cache

[4K I/O Buffer Pool]
  wash size = 4096 K

[Meta-Data Caches]
  number of open objects = 2000
  number of open indexes = 200
  number of open partitions = 750

[Disk I/O]
  disk i/o structures = 1024
  number of devices = 127

[Unicode]
  enable unicode normalization = 0

[Network Communication]
  default network packet size = 4096
  max network packet size = 4096

[O/S Resources]
  max async i/os per engine = 2048
  max async i/os per server = 2048

[Query Tuning]
  number of worker processes = 140
  max parallel degree = 140
  max scan parallel degree = 6

[Physical Memory]
  max memory = 13522443
  additional network memory = 1288192
  lock shared memory = 1
  heap memory per user = 1600

[Processors]
  max online engines = 24
  number of engines at startup = 2

[SQL Server Administration]
  procedure cache size = 200000
  cpu accounting flush interval = 100000
  i/o accounting flush interval = 100000
  runnable process search count = 2
  housekeeper free write percent = 0
  enable housekeeper GC = 0
  i/o batch size = 16

[User Environment]
  number of user connections = 450

[Lock Manager]
  lock address spinlock ratio = 50

[Java Services]
  size of process object heap = 150

```

SYBASE, INC.
WORLDWIDE HEADQUARTERS
ONE SYBASE DRIVE
DUBLIN, CA 94568-7902 USA
Tel: 1 800 8 SYBASE

www.sybase.com

Copyright © 2009 Sybase, Inc. All rights reserved. Unpublished rights reserved under U.S. copyright laws. Sybase, and the Sybase logo are trademarks of Sybase, Inc. or its subsidiaries. All other trademarks are the property of their respective owners. ® indicates registration in the United States. Specifications are subject to change without notice. 3/09.

SYBASE®