

Drupal Performance Improvement via SSD Technology

A Sun ISV Engineering Test Report

Eric R. Reid, Sun Microsystems, Inc.
Draft 1.1 of 8 March 2009

Introduction

Drupal is a leading Open Source Content Management System, written in PHP for the AMP stack. While Drupal does not usually place an extremely heavy burden on I/O subsystems 'out of the box', Sun expects that the introduction of Solid State Disk (SSD) Technology in upcoming server products will provide performance increases. This document details the testing performed to identify and quantify such increases.

Drupal 6 was to be tested, using a Sun-developed synthetic workload, on Solaris 'Nevada' and the Coolstack AMP stack. Configurations would be created and measured which provided three types of I/O subsystem support:

1. Traditional internal Hard Disk Drive (HDD) storage
2. New internal SSD devices as whole, functionally identical replacements for HDDs
3. Utilizing ZFS Hybrid Storage Pools (HSPs), which effectively mix SDD and HDD technologies (reference: <http://blogs.sun.com/brendan/entry/test>)

Testing Approach

As Drupal is usually run 'out of the box' on lightly-tuned single servers, we configured a single Sun CoolThreads server with Drupal, Coolstack (including PHP, Apache Webserver and MySQL DBMS) on Solaris Nevada. ZFS was used for all database filesystems (where the predominance of I/O takes place in Drupal-based systems). Filesystems were created for internal HDD-based ZFS pools, as well as SSD-based pools and HSP pools. Performance monitoring was done for both Database Generation (mostly MySQL writes) and Drupal Workload tests. Workloads were generated from a separate server.

Benchmark Configuration

	Drupal Server	Load Generation Server
Server Hardware	Sun SPARC Enterprise T5140 <ul style="list-style-type: none">• 2 2 UltraSPARC-T2 Plus processors @ 1.2GHz (total 16 cores, 128 threads)• 64GB RAM	Sun Fire x4200 M2 (Load Generation) <ul style="list-style-type: none">• 2 dual-core AMD Opteron Processors @ 2.8 GHz (total 4 cores)• 8GB RAM

	Drupal Server	Load Generation Server
	<ul style="list-style-type: none"> • 3 146GB 10K RPM Internal SAS Disks • 4 32GB SSD Disks • 10/100/1000Mbps Ethernet 	<ul style="list-style-type: none"> • 10/100/1000Mbps Ethernet
Operating System	Solaris 'Nevada' Build 108	Solaris 10 10/06
AMP Stack	CoolStack 1.3.1: <ul style="list-style-type: none"> • MySQL 5.1.25 32-bit • Apache 2.2.9 • PHP 5.2.6 • APC 3.0.19 	JVM 1.5.0_07
Application	Drupal 6.9	Custom load generator based on Faban 0.9

Software Customization

Software	Customization
Solaris	None
MySQL	INNODB tables standard my.cnf based on my.innodb-heavy-4G.cnf
Apache	Pointed \$DOCROOT at Drupal 6
APC	Enabled
PHP	None
Drupal	Normal Caching enabled Page Compression enabled Block Cache enabled Optimize CSS files enabled Optimize Javascript files enabled

Please see Appendix for configuration file detail.

The Drupal Workloads

No standardized Drupal workload exists in nature, and as such Sun has undertaken the development of a synthetic workload to simulate a simplistic Drupal user workload and database. This Drupal instance consists of:

- A 'Devel' module-created database (no custom or added modules) with 100K nodes (pages) and 10K users; this database was created with INNODB tables.
- 'Clean URLs' must be enabled in Drupal for this database to work properly

- A 'Web101'-based faban driver, written with a mix of eight common Drupal operations that can be performed by anonymous visitors, the Admin user, or registered users to the site:
 - FP: Access front page
 - RP: Access random page
 - RN: Access random node
 - UL: Random user login
 - AL: Admin login
 - CS: Logged-in user Create Story
 - CP: Logged-in user Create Page
 - LO: Logout

The state-change matrix chosen:

From/To	FP	RP	RN	UL	AL	CP	CS	LO
FP	20	40	20	15	5	0	0	0
RP	40	20	20	15	5	0	0	0
RN	20	20	40	15	5	0	0	0
UL	0	0	0	0	0	40	30	30
AL	0	0	0	0	0	10	10	80
CP	0	0	0	0	0	0	0	100
CS	0	0	0	0	0	0	0	100
LO	20	40	20	15	5	0	0	0

Test Runs

A test run consists of some number of simultaneous synthetic Drupal users fired up on the load generation server. Each 'user' generates a workload according to the state matrix above, and sends HTTP requests to the Drupal server. Each run ramps up for 120 seconds, maintains steady state for 900 seconds, ramps down for 120 seconds. A 300 second pause is inserted between runs.

The faban harness measures ops/sec and response time across all operations. In addition, Solaris and MySQL statistics are gathered during the runs.

Three sets of runs were conducted, with the number of concurrent users ramped up from 1 to over 400. Linear scalability, throughput (in ops/sec) and response time were noted; at some point in each run set, performance fell off due to resource exhaustion or failure to scale of the underlying software stack.

The three sets of runs varied the nature of the storage used for the underlying MySQL data and logs:

1. A ZFS pool (16K recordsize) on a two internal 10K RPM SAS HDDs
2. A ZFS pool (16K recordsize) across two internal SSDs

3. A Hybrid ZFS Storage Pool (16K recordsize) using one internal SSD as Cache and a single internal SAS HDD as Primary

Before each run was conducted, the SQL script used to create the test database was run and timed. This .sql script was comprised of approximately 52 TABLE CREATE and 1004 INSERT statements.

Observed Test Results and Analysis

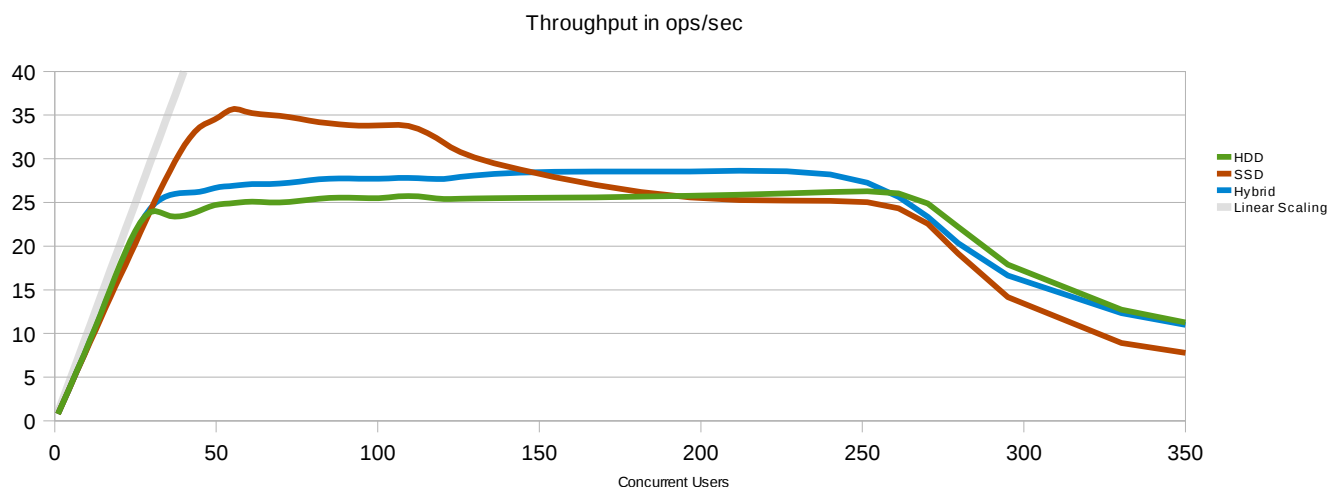
A. Database Creation

Storage Configuration	Wall Time to Create Database
ZFS pool – 2 10K RPM SAS HDD	54m 35.097s
ZFS pool – 2 SSDs	53m 0.875s (3.0% improvement)
Hybrid Storage Pool - Cache: 1 x SSD Primary: 1 x HDD	53m 27.649s (2.1% improvement)

Minimal performance improvement of this SQL database creation script was observed. Given the nature of the script – mostly CREATEs and INSERTs into InnoDB tables – this is not surprising. SQL and MySQL tuning could probably improve these numbers, but this was not a goal of this exercise.

B. Drupal Runs

Storage Configuration	System keeps up with demand (near-linear scaling)	Max Observed Ops/sec	'403' Errors Observed
ZFS pool – 2 10K RPM SAS HDD	Up to 30 users	26.5 ops/sec	At and above 110 users
ZFS pool – 2 SSDs	Up to 45 users (50% better)	36.2 ops/sec (36% better)	At and above 110 users
Hybrid Storage Pool - Cache: 1 x SSD Primary: 1 x HDD	Up to 30 users	29.0 ops/sec (9% better)	At and above 110 users



With the Drupal Workload we use for performance testing (see the mix above), significant performance increases (30-50%) were observed with substituting SSDs for HDDs. In addition, an HSP using one of each type disk showed slightly better performance over HDD, but actually provided slightly better throughput between 100 and 250 users. We suspect that ZFS and MySQL tuning would be required to show even greater performance improvements.

It should be noted that CPU cycles were never an issue in these tests – at maximum during any runs, no more than 8% total CPU utilization across the 128 hardware CMT threads was observed. Memory was also never constrained during any run, nor were any of the Zpools in these tests (max 10% busy).

The question this raises, then, is 'what exactly constrains this stack?' As we have seen in similar testing of Drupal/AMP before, MySQL and Apache HTTPD Server do not necessarily scale well 'out of the box' without significant platform-specific tuning. A well-tuned stack will stress the devices to a greater degree, and we suspect relative performance numbers to be similar.

As noted in the table above, Web Server '403' errors are observed starting at 110 simultaneous users. If we consider that the 'saturation' point for this configuration, we make note of the throughput at 110 users:

Storage Configuration	Throughput @ 110 users
ZFS pool – 2 10K RPM SAS HDD	25.96 ops/sec
ZFS pool – 2 SSDs	34.12 ops/sec (31% better)
Hybrid Storage Pool - Cache: 1 x SSD Primary: 1 x HDD	27.93 ops/sec (8% better)

Conclusions

SSDs do provide apples-to-apples performance improvement over HDDs (even 10K RPM drives). One challenge with AMP-based packages such as Drupal is that they are not often I/O-bound unless precisely tuned for particular workloads. As such, 'out of the box' performance gains might not justify the increased cost.

Properly-tuned Drupal deployments which stress I/O subsystems will likely see greater SSD-vs-HDD performance increases than detailed in this report.

ZFS Hybrid Storage Pools provide a more cost-effective option in which some SSD speed advantages can be realized without the full upcharge of SSDs.

Appendix – Disk Devices

```
bash-3.2# format
Searching for disks...done
c2t0d1: configured with capacity of 407.98GB
c2t0d31: configured with capacity of 16.00MB
AVAILABLE DISK SELECTIONS:
  0. c1t0d0 <SUN146G cyl 14087 alt 2 hd 24 sec 848>
    /pci@400/pci@0/pci@8/scsi@0/sd@0,0
  1. c1t1d0 <SEAGATE-ST914602SSUN146G-0400-136.73GB>
    /pci@400/pci@0/pci@8/scsi@0/sd@1,0
  2. c1t2d0 <ATA-INTEL SSDSA2SH03-8620-29.80GB>
    /pci@400/pci@0/pci@8/scsi@0/sd@2,0
  3. c1t3d0 <ATA-INTEL SSDSA2SH03-8620-29.80GB>
    /pci@400/pci@0/pci@8/scsi@0/sd@3,0
  4. c1t4d0 <ATA-INTEL SSDSA2SH03-8620-29.80GB>
    /pci@400/pci@0/pci@8/scsi@0/sd@4,0
  5. c1t5d0 <ATA-INTEL SSDSA2SH03-8620-29.80GB>
    /pci@400/pci@0/pci@8/scsi@0/sd@5,0
  6. c1t1d0 <SEAGATE-ST914602SSUN146G-0400-136.73GB>
    /pci@400/pci@0/pci@8/scsi@0/sd@6,0
  7. c2t0d0 <SUN-LCSM100_F-0670-408.00GB>
    /pci@500/pci@0/pci@9/SUNW,qlc@0/fp@0,0/ssd@w202200a0b8389
2ec,0
  8. c2t0d1 <SUN-LCSM100_F-0670 cyl 52222 alt 2 hd 256 sec 64>
    /pci@500/pci@0/pci@9/SUNW,qlc@0/fp@0,0/ssd@w202200a0b8389
2ec,1
  9. c2t0d31 <SUN-UniversalXport-0670 cyl 8 alt 2 hd 64 sec 64>
    /pci@500/pci@0/pci@9/SUNW,qlc@0/fp@0,0/ssd@w202200a0b8389
2ec,1f

# Run set 1: two HDD drives
bash-3.2# zpool create db_hdd c1t1d0 c1t6d0
bash-3.2# zfs set recordsize=16k db_hdd

# Run set 2: two SSD drives
bash-3.2# zpool create db_ssd c1t2d0 c1t3d0
bash-3.2# zfs set recordsize=16k db_ssd

# Run set 3: one 146GB HDD primary, one 32GB SSD cache
bash-3.2# zpool create db_hybrid c1t1d0
bash-3.2# zpool add db_hybrid cache c1t2d0
bash-3.2# zfs set recordsize=16k db_hybrid
```

Appendix – Tuning Options

MySQL 5.1.25 (32-bit)

```
[client]
port          = 3306
socket        = /tmp/mysql.sock

[mysqld]
port          = 3306
socket        = /tmp/mysql.sock
datadir       = /db_hybrid
back_log      = 50
max_connections = 100
max_connect_errors = 10
table_cache  = 2048
max_allowed_packet = 16M
binlog_cache_size = 1M
max_heap_table_size = 64M
sort_buffer_size = 8M
join_buffer_size = 8M
thread_cache_size = 8
thread_concurrency = 8
query_cache_size = 64M
query_cache_limit = 2M
ft_min_word_len = 4
default_table_type = MYISAM
thread_stack = 192K
transaction_isolation = REPEATABLE-READ
tmp_table_size = 64M
log-bin=mysql-bin
log_slow_queries
long_query_time = 2
log_long_format
server-id = 1
key_buffer_size = 32M
read_buffer_size = 2M
read_rnd_buffer_size = 16M
bulk_insert_buffer_size = 64M
myisam_sort_buffer_size = 128M
myisam_max_sort_file_size = 10G
myisam_max_extra_sort_file_size = 10G
myisam_repair_threads = 1
myisam_recover
innodb_additional_mem_pool_size = 16M
innodb_buffer_pool_size = 2G
innodb_data_file_path = ibdata1:800M:autoextend
innodb_file_io_threads = 4
innodb_thread_concurrency = 16
innodb_flush_log_at_trx_commit = 1
```

```
innodb_log_buffer_size = 8M
innodb_log_file_size = 256M
innodb_log_files_in_group = 3
innodb_max_dirty_pages_pct = 90
innodb_lock_wait_timeout = 120
```

```
[mysqldump]
```

```
quick
```

```
max_allowed_packet = 16M
```

```
[mysql]
```

```
no-auto-rehash
```

```
[isamchk]
```

```
key_buffer = 512M
```

```
sort_buffer_size = 512M
```

```
read_buffer = 8M
```

```
write_buffer = 8M
```

```
[myisamchk]
```

```
key_buffer = 512M
```

```
sort_buffer_size = 512M
```

```
read_buffer = 8M
```

```
write_buffer = 8M
```

```
[mysqlhotcopy]
```

```
interactive-timeout
```

```
[mysqld_safe]
```

```
open-files-limit = 8192
```