

TEXAS MEMORY SYSTEMS, INC.

RamSan-500

Integration Guide



Version 1.0

Any trademarks or registered trademarks used in this document belong to the companies that own them.

Copyright © 2009, Texas Memory Systems, Inc. All rights are reserved. No part of this work may be reproduced or used in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without permission of the copyright owner.

Table of Contents

Preface	1
Document Overview	1
Revision History	1
Chapter 1 – Introduction.....	2
Chapter 2 – Multipathing Best Practices	3
2.1 RamSan FC Controllers	3
2.2 Host Bus Adapter	3
2.3 Cabling.....	4
2.4 Zoning.....	4
2.5 LUN Masking	4
2.6 Multipathing Setup.....	4
2.6.1 Windows	5
2.6.2 Linux	5
2.6.3 Solaris 8/9	5
2.6.4 Solaris 10	6
2.6.5 Veritas/Symantec Storage Foundation.....	6
Chapter 3 – Filesystem Alignment	7
3.1 Microsoft Windows Operating Systems	7
3.2 Linux	8
3.3 Confirming alignment.....	9
Chapter 4 – Cache Configuration	10
4.1 Read Ahead.....	10
4.2 Sync Always	11
4.3 Dirty Cache Throttle	11
Chapter 5 – Locked LUN	13
Chapter 6 – Summary	14
Appendix A – multipath.conf	15

Preface

Document Overview

This document details how to get the maximum performance from the RamSan-500. This guide will cover the best practices for redundancy as well as performance in single-server and multi-server environments.

Revision History

The following table describes revisions to this document:

Version	Comments	Date
1.0	Initial release	1/15/2009

Chapter 1 – Introduction

Texas Memory Systems designed its Solid State Disk (SSD) products using standard interfaces that support a wide range of servers. Servers are generally configured to work around the performance limitations of Hard Disk Drives (HDD) and some tweaks can help take advantage of the speed of SSDs.

This document goes through Multipathing best practices, RamSan-500 cache configuration, and alignment strategies to reach the optimal performance from the RamSan-500. Additional information can be obtained by contacting support@ramsan.com.

Chapter 2 – Multipathing Best Practices

Multipath Input/Output technology enhances both redundancy and performance. Each Fibre Channel (FC) port on the RamSan acts as an independent access point to its assigned volumes. This allows the host to actively utilize multiple active channels to each volume, resulting in higher bandwidth and resiliency against path loss.

2.1 RamSan FC Controllers

Each dual ported 4Gb Fibre Channel controller in a RamSan-500 provides bandwidth of close to 800MBps. When utilizing 4 FC controllers, there is an aggregate bandwidth of 3.2GBps of cached read or write bandwidth to a RamSan-500. On 100% random, cache-miss I/O, the maximum sustained bandwidth is 2GBps. For most application workloads, however, the cache plays a very significant role.

A single optical transceiver is the most likely component to experience failure on an FC controller. There are two transceivers on a FC controller, so a single transceiver does not fault the entire card. However, both ports do share a common processor. To eliminate any FC failure point multipathing is recommended to span across two or more controllers before utilizing the second ports on the controllers. For performance, all IOPS and bandwidth scale with the number of controllers used until the RamSan-500 performance limits are met.

The RamSan-500 allows very flexible presentation of LUNs through its FC controllers. A LUN can be presented and accessed through every port simultaneously. The loss of one controller does not affect the I/O on another controller. This is commonly referred to as supporting Active/Active or Active/Active symmetric multipathing.

Almost all major operating systems incorporate multipathing. If you have questions about multipathing not covered in this document, please contact RamSan support at support@ramsan.com.

2.2 Host Bus Adapter

Texas Memory Systems recommends using dual-ported HBAs when possible. Dual-ported HBAs provide additional ports for aggregating bandwidth while conserving the limited expansion slots on most servers. In an ideal case, multiple dual-ported HBAs are used within each server for redundancy. RamSans are tested for interoperability against all major HBA vendors. If you have any questions about support for a particular HBA, please contact RamSan support at support@ramsan.com.

2.3 Cabling

Cabling should be designed to provide resiliency across RamSan ports, FC controllers, switches, server HBA ports, and HBAs. All high availability (HA) concepts of a dual-fabric setup apply to the RamSan. Since the RamSan supports Active/Active multipathing, load-balancing does not have to be manually architected. Instead, all I/O can be sent across both sides of the fabric equally.

One key element to recognize when cabling a RamSan is the use of available paths. In order to take advantage of RamSan ports, there must be an equal number of server ports. Otherwise, there are underutilized ports on one side of the fabric.

The RamSan is built to deliver I/O through all connected ports. A RamSan-500 supports up to 4 FC controllers and a total 8 available ports. Using all 8 ports is recommended if the application can benefit from additional bandwidth.

2.4 Zoning

In a switched fabric deployment, it is a common practice to isolate one application's Server-Storage devices from other applications' Server-Storage devices to prevent cross traffic and help ease maintenance. TMS recommends zoning in all multi-server environments.

Zoning is best deployed in a 1:1 port ratio with 1 server's HBA port for 1 RamSan FC port. A server HBA port serving I/O to multiple FC ports within the same RamSan results in excessive paths since the server's single HBA port is the limiting factor for performance.

2.5 LUN Masking

LUN Masking is a utility that allows separate servers to access separate LUNs through a common controller port. It is possible to use LUN Masking in place of zoning, but it is better utilized as a supplement to zoning. If multiple servers access different LUNs through a common RamSan port, then LUN Masking can be used to only present the LUNs to the server that needs access. This is not a common configuration since it is a best practice to have each server access the RamSan through isolated ports for performance reasons.

2.6 Multipathing Setup

Almost every major operating system has a built in multipathing option to prevent loss of service when a path to a disk is lost. Each OS has different parameters, and some require additional software to take advantage of these features. When deploying a RamSan into a multipathed environment, be sure to include path failure in the test plan prior to deploying into production service.

2.6.1 Windows

Windows 2003 has a built-in MPIO driver provided by Microsoft. This driver is responsible for aggregating the links of RamSan systems and reporting the addition or removal of links while online to the kernel. In order to utilize this feature with a RamSan, a Device Specific Module (DSM) is provided by Texas Memory Systems to alert the MPIO driver of the RamSan's identity and its Active/Active capabilities. This driver is available free upon request to support@ramsan.com.

If running EMC PowerPath, you must upgrade to 4.6 or later before using the RamSan DSM. All major storage vendors have support for the MPIO multipathing and coexist safely due to the common driver in Windows.

Windows 2008 no longer requires a DSM provided by Texas Memory Systems. Instead, the Multipath Feature option must be installed on the 2008 server. From the Control Panel applet, the attached RamSan can be selected and approved for multipathing. After reboot, all RamSan storage will have full multipathing support. The Round-robin, or Lowest Queued Path, policy should be applied on the RamSan multipath device to take advantage of higher performance. This can be done through the Properties dialog in Device Manager.

2.6.2 Linux

Linux kernels on 2.6 and higher support multipathing through *device-mapper-multipath*. This package can coexist with other multipathing solutions as long as the other storage is excluded from *device-mapper*. Appendix A includes a template for the *multipath.conf* file for utilizing the RamSan to its full potential.

Since the RamSan controllers provide true Active/Active I/O, the *rr_min_io* field in the *multipath.conf* file is set to 1. This results in a round-robin distribution of I/O across all available paths.

To further improve the performance on Linux, append the kernel parameter "elevator=noop" to disable the I/O scheduler. This will help reduce latency on small-block requests to the RamSan.

2.6.3 Solaris 8/9

MPxIO is the built-in multipathing mechanism for Solaris and requires a Sun-branded HBA. When using a non-Sun-branded HBA, then multipathing must be achieved by Veritas Storage Foundation.

To enable MPxIO support for RamSans, modify the */kernel/drv/scsi_vhci.conf* file with the following:

```
load-balance="round-robin";
mpxio-disable="no";

device-type-scsi-options-list =
"TMS      RamSan", "symmetric-option";
symmetric-option = 0x1000000;
```

2.6.4 Solaris 10

Just as Solaris 8/9, MPxIO is the built-in multipathing mechanism for Solaris and requires a Sun-branded HBA. When using a non-Sun-branded HBA, multipathing can be achieved by Veritas Storage Foundation.

To enable MPxIO support for RamSans, modify the /kernel/drv/scsi_vhci.conf file with the following:

```
load-balance="round-robin";

device-type-scsi-options-list =
"TMS      RamSan", "symmetric-option";
symmetric-option = 0x1000000;
```

In addition, the /kernel/drv/fp.conf will require:

```
mpxio-disable="no";
```

2.6.5 Veritas/Symantec Storage Foundation

Veritas/Symantec provides an array support library (ASL) for RamSans on Solaris Sparc systems. This ASL will work with all RamSans.

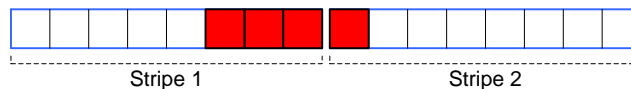
Veritas/Symantec also has "just a bunch of disks" (JBOD) support for RamSans but requires the following command to acknowledge multiple RamSan LUNs. Run this command on the host.

```
vxddladm addjbod vid="TMS" length=10
```

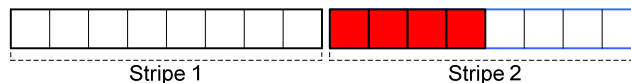
After which, Storage Administrator will list the RamSan LUNs appropriately.

Chapter 3 – Filesystem Alignment

Filesystem alignment is a technique to match the filesystem I/O requests with important block boundaries in the physical storage system. This is important in any system that implements a RAID layout, as I/O requests that fall within the bounds of a single stripe will have better performance than an I/O request that crosses over multiple stripes. When an I/O occurs across the end-point of one stripe and into another, the controller must modify both stripes to maintain their consistency. This is visually depicted in the following graphic:



When the I/O is aligned, the result is a single modified stripe.



For the RamSan-500, alignment is optimal at 16KB.

3.1 Microsoft Windows Operating Systems

By default, Windows will offset the partition by 63 sectors, or 32.5KB. To align to the preferred 16KB, the offset must be done using the diskpart.exe utility.

First, select the disk (LUN) that will hold the filesystem:

```
DISKPART> list disk

Disk ###  Status   Size     Free     Dyn  Gpt
-----  -
Disk 0    Online   69 GB    0 B
Disk 1    Online   90 GB    0 B
Disk 2    Online   10 GB    0 B
Disk 3    Online   10 GB    10 GB

DISKPART> select disk 3

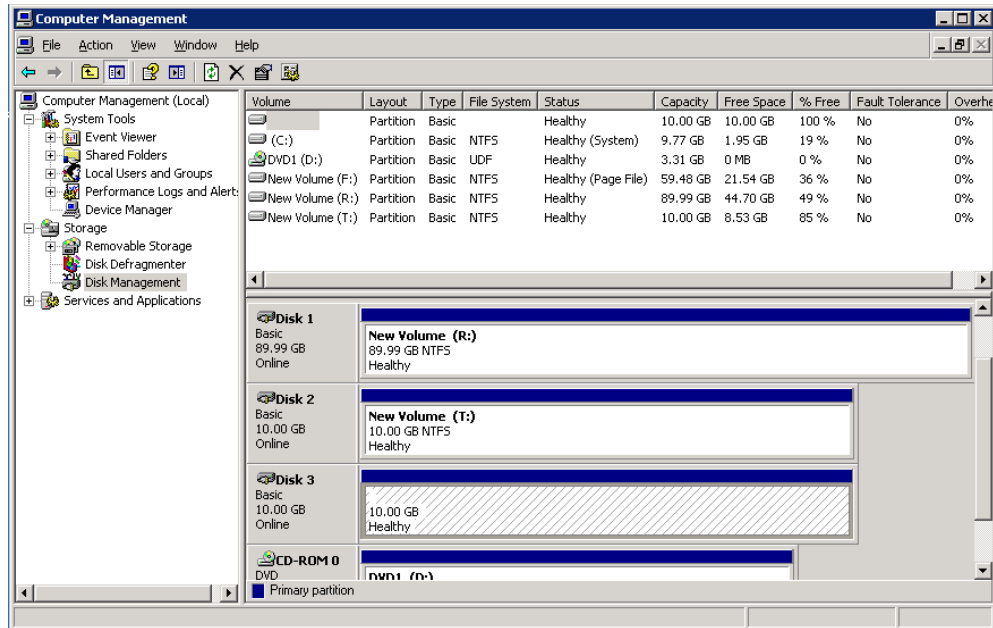
Disk 3 is now the selected disk.
```

Once selected, a partition can be made on the disk.

```
DISKPART> create partition primary align=64

DiskPart succeeded in creating the specified partition.
```

Now, a filesystem or drive letter (RAW access) can be assigned to the aligned partition for application use.



3.2 Linux

Linux defaults to a 63-sector offset. To align a partition in Linux, use fdisk and do the following:

```
# fdisk /dev/mapper/ramsan
Command (m for help): u
Changing display/entry units to sectors

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First sector (63-16777215, default 63): 128
Last sector or +size or +sizeM or +sizeK (128-16777215,
default 16777215):
Using default value 16777215

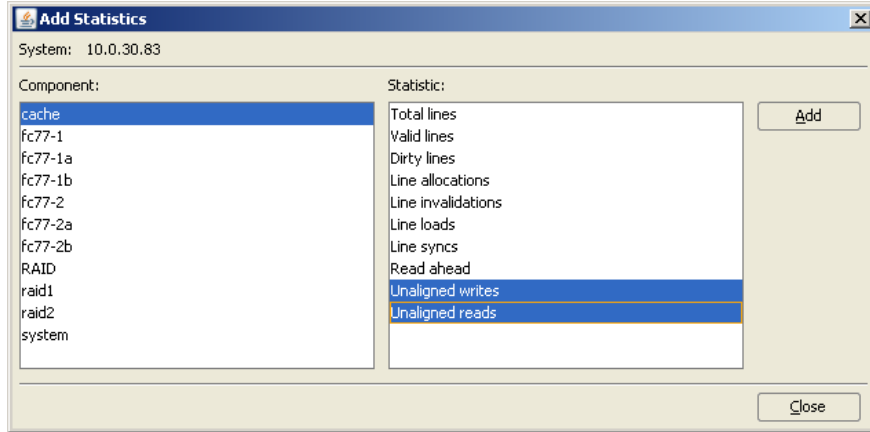
Command (m for help): w
The partition table has been altered!
```

The newly created partition now has an offset of 64KB and will perform optimally with an aligned application.

3.3 Confirming alignment

Confirming alignment can be accomplished by monitoring two counters in the RamSan-500's Web GUI.

In the Statistics Tree item, add the Cache=>Unaligned Writes and Cache=>Unaligned Reads counters.

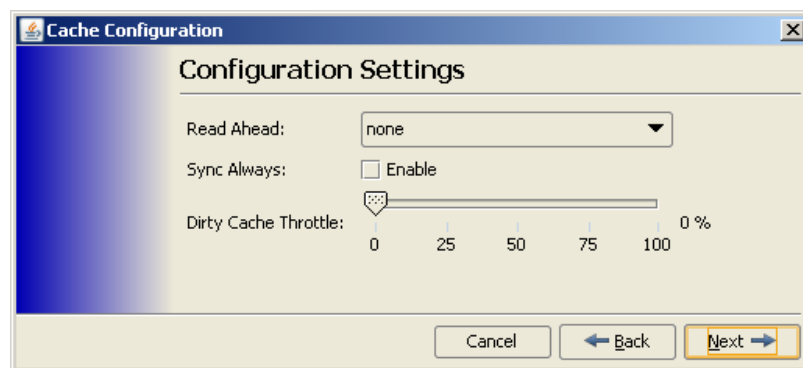


These counters represent unaligned accesses to the RAID. These are running counters that can easily be monitored during active RamSan-500 use to determine if I/O accesses are aligned. If the counters are only slightly increasing, then I/O is aligned. If the counters are increasing at a much higher rate (hundreds or more per second) then I/O is not aligned.

Chapter 4 – Cache Configuration

The cache in the RamSan-500 is primarily a benefit to write performance, as the native performance of enterprise Flash is outstanding on read I/O. The RamSan-500 has redundant internal batteries with enough power to keep the system running long enough to synchronize all of the Cache without external power. This allows the cache to hold dirty data. Data is considered “dirty” if it has changed in the cache but the change has not been sent to the storage volume.

The cache has a few tunable options that can be modified by the dialog box shown below. This dialog box is accessible through the Web GUI. In the Storage=>Cache tree item click the “Configure” toolbar button to reach this screen.



The default cache options are disabled Sync Always and 0% Dirty Cache Throttle. This is designed to flush the cache as quickly as possible to the Flash RAID. With Read Ahead set to “none”, all read I/O will be without overhead.

4.1 Read Ahead

The Read Ahead option can be generally disregarded. This option has a global scope to all Logical Units and will enforce a larger read size for each read to the Flash RAID. This allows small block sequential reads to be available at close to DRAM speeds, but random read I/O will return with slightly higher latencies. Even though the DRAM cache read is 10x faster than the Flash read, TMS recommends leaving this option set to “none” except in conditions where the application does pure sequential reads.

For the read ahead option, the incremental values are quantified as none (0), low (16KB), medium (32KB), high (64KB), and max (512KB).

4.2 Sync Always

Sync Always is used in conjunction with the Dirty Cache Throttle setting. Enabling this feature instructs the cache to always synchronize data that is in the cache to the RAID. This feature will proactively write data from the cache to the Flash RAID on at a low priority to reduce the amount of contention between reads and writes on the RAID. Upon a write to the RAID, the cache-line is not evicted (dropped from the cache), but simply marked clean and left available for a cache-hit on the next request.

By keeping the cache in sync, massive bursts of writes can be handled by simply evicting all clean lines in the cache without any impact to the Flash RAID performance.

4.3 Dirty Cache Throttle

The Dirty Cache Throttle option determines how aggressively the cache synchronizes dirty data to the RAID. This setting is set as a percentage between 0% and 100%. When the amount of dirty data in the cache is below the percent selected by this setting, data from the cache is only passively synchronized to the RAID if the Sync Always box is checked. Once the amount of dirty data in the cache exceeds this percentage, the dirty data is more aggressively synchronized.

The default setting of 0% indicates that writes will continuously be synced to the RAID as a background process. A setting of 100% will allow the entire cache to hold written data therefore accelerating further writes to the same cache lines. Under this scenario, recently written cache lines are not persisted to the RAID until necessary and heavy writes to cache will not cause contention to the RAID as the I/O will be served repeatedly by the cache.

The drawback to maximizing the Dirty Cache Throttle is how the system will perform under a burst of random writes. If the cache is 100% dirty, then before each write the cache must synchronize to evict a dirty cache line.

Leaving the Dirty Cache Throttle at 0% increases the contention on the RAID as multiple writes to the same cache line are more likely to result in multiple writes to the RAID. This allows larger bursts of writes to be serviced without sacrificing performance. Even with the slight contention, the amount of write performance available still exceeds most application capabilities.

When writes to the RamSan begin filling the cache beyond the threshold set for the Dirty Cache Throttle, the RamSan will become more aggressive in destaging the writes to the Flash RAID. As the cache becomes more consistent with data on the Flash RAID, the writes will become less aggressive. When the written data in the cache falls below the Dirty Cache Throttle, data is only written to the RAID if the Sync Always option is enabled, and even then only under the lowest of priorities. After data in the cache has been committed to the Flash RAID, the cache line remains to service any subsequent I/O at DRAM speed.

Chapter 5 – Locked LUN

If a Logical Unit is smaller than ½ the capacity of the DRAM cache, then the “Turbo” option is available to lock the LUN in cache. Once a LUN is locked in cache, any data that belongs to this LUN will no longer be evicted from cache. This means that an asynchronous mirror will form between DRAM and Flash for the LUN. This option is intended for extremely heavy I/O volumes or volumes that benefit from the performance of DRAM.

Systems are limited to a single Locked LUN less than 50% of cache capacity. On a RamSan-500 with 64GB of cache, a 32GB LUN can be locked.

Chapter 6 – Summary

The performance tweaks and best practices covered in this document will help get the highest performance from the RamSan-500. Some of the configuration settings depend on the I/O profile of the application using the RamSan-500. Questions regarding analysis and tuning for each application are welcome and encouraged: support@ramsan.com.

Appendix A – multipath.conf

```
defaults {
    udev_dir                /dev
    polling_interval        10
    selector                "round-robin 0"
    path_grouping_policy    multibus
    getuid_callout          "/sbin/scsi_id -g -u -s /block/%n"
    prio_callout            /bin/true
    path_checker            readsector0
    rr_min_io               1
    rr_weight               priorities
    failback                immediate
    no_path_retry           fail
    user_friendly_name      yes
}
devnode_blacklist {
#    wwid 01234567890123456
#    wwid "1234*"
    devnode "^(ram|raw|loop|fd|md|dm-|sr|scd|st)[0-9]*"
    devnode "^hd[a-z]"
    devnode "^cciss!c[0-9]d[0-9]*"
    devnode "^sd[a]"
}
multipaths {
    multipath {
        #wwid of RamSan - retrieved by running the getuid_callout above
        #scsi_id -g -u -s /block/sdb returns the wwid of 20020c24000036071

        wwid                20020c24000036071
        alias                ramsan

        # This RamSan volume is accessible now as /dev/mapper/ramsan
    }
}
devices {
    device {
        vendor                "TMS"
        product                "RamSan"
    }
}
}
```