

## Blueprints : Configuring Sun Storage J4000 Arrays and the ZFS File System in Ten Minutes

This page last changed on Feb 06, 2009 by [elena\\_levashova](#).

by Dominic Kay  
July, 2008

[ [Introduction](#) ] [ [Building the ZFS Pool](#) ] [ [Introduction](#) ] [ [Maximum Space - A Simple Pool](#) ] [ [Maximum Safety - Mirroring The Two Arrays](#) ] [ [Space/Speed Compromise - RAIDZ](#) ] [ [Configuring a Single Array - RAIDZ](#) ] [ [Creating Filesystems & Backing them Up](#) ] [ [Creating a File System](#) ] [ [Backup: Taking a Snapshot](#) ] [ [Saving and Sending A Snapshot](#) ] [ [Conclusion](#) ] [ [Finding out More](#) ]

### Introduction

The purpose of this short guide is to get the reader up and running with the ZFS file system and Sun Storage J4000 array. We will cover ZFS pools and file systems, the different RAID Levels such as Mirroring (between JBODS) and RAIDZ. We will demonstrate hot sparing, snapshots and online backups.

Assembling volumes, building file systems on them, creating snapshots and backing them up would traditionally all require separate pieces of software. However, the ZFS file system provides a "one-stop shop" for all these tasks and it is our aim to show how amazingly easy it is to accomplish them. While there is a Graphical User Interface provided with the ZFS file system to address critical tasks, the ZFS file system is simple enough to accomplish them without it.

The reader is probably a field engineer, system architect or technical decision maker. A little general Unix knowledge is assumed but you don't need to be a seasoned Solaris administrator to understand this guide. Its a short read so links are provided to further reading at the end.

The examples in this document were produced on a Sun Fire X4150 running OpenSolaris 2008.05, connected to two Sun Storage J4200 arrays, each attached to independent PCIe SAS HBA cards. In our example we use 146 GB SAS HDDs in our arrays, but the Storage J4200 also supports SATA HDDs - the syntax would remain the same.

### Building the ZFS Pool

#### Introduction

The first step is to put all the disks in the arrays into a ZFS pool from which we can then allocate file systems. We have a number of configuration possibilities here but two examples will get you started.

Our two arrays each contain 12 x 136.72 GB disks - approximately 3.2 TB.

In our examples we will call our pool mypool and you can remove it if you need it using

```
# zpool destroy mypool
```

to try out your different options.

#### Maximum Space - A Simple Pool

For some purposes it might be acceptable to have no contingency in case of disk failure at all - e.g this may be a testing area or scratch space in a Test & Development environment. The simplest arrangement here would be to put all the disks in one pool:

```
# zpool create mypool c6t0d0 c6t1d0 c6t2d0 c6t3d0 c6t4d0 c6t5d0 c6t6d0 c6t7d0 c6t8d0 c6t9d0 c6t10d0 c6t11d0 c7t0d0 c7t1d0 c7t2d0 c7t3d0 c7t4d0 c7t5d0 c7t6d0 c7t7d0 c7t8d0 c7t9d0 c7t10d0 c7t11d0
```

ZFS likes whole disks - it (re)partitions and (re)labels disks when you bring them into a ZFS storage pool. We can show our pool using the list subcommand:

```
# zpool list mypool
NAME  SIZE  USED  AVAIL  CAP  HEALTH  ALTROOT
mypool 3.19T 129K 3.19T  0%  ONLINE  -
```

This option gives us just over 3 Tb of useable space. We can dig deeper into the configuration of the pool using the status subcommand:

```
# zpool status mypool
pool: mypool
state: ONLINE
scrub: none requested
config:

   NAME      STATE  READ WRITE CKSUM
   mypool    ONLINE  0   0   0
   c6t0d0    ONLINE  0   0   0
   c6t1d0    ONLINE  0   0   0
   ....
   c6t11d0   ONLINE  0   0   0
   c7t0d0    ONLINE  0   0   0
   c7t1d0    ONLINE  0   0   0
   ....
   c7t11d0   ONLINE  0   0   0

errors: No known data errors
```

### Maximum Safety - Mirroring The Two Arrays

For maximum resilience we can mirror our two arrays and provide hot spares in case of component failure as follows:

```
# zpool create mypool mirror c6t0d0 c7t0d0 mirror c6t1d0 c7t1d0 mirror c6t2d0 c7t2d0 mirror c6t3d0 c7t3d0 mirror c6t4d0
c7t4d0 mirror c6t5d0 c7t5d0 mirror c6t6d0 c7t6d0 mirror c6t7d0 c7t7d0 mirror c6t8d0 c7t8d0 mirror c6t9d0 c7t9d0 mirror
c6t10d0 c7t10d0
# zpool add mypool spare c6t11d0 c7t11d0
```

This creates individual mirrors of paired disks. The data will be striped across the top of them. Also one hotspare in each array.

```
# zpool list mypool
NAME  SIZE  USED  AVAIL  CAP  HEALTH  ALTROOT
mypool 1.46T 136K 1.46T  0%  ONLINE  -
```

This gives us just under half our total space.

### Space/Speed Compromise - RAIDZ

RAID-Z is the technology used by ZFS to implement a data-protection scheme which is less costly than mirroring in terms of storage overhead.

There is a performance penalty to pay for the increased amount of storage available but it is often acceptable. Such a pool is created as follows:

```
# zpool create mypool raidz c6t0d0 c7t0d0 c6t1d0 c7t1d0 c6t2d0 c7t2d0 c6t3d0 c7t3d0 c6t4d0 c7t4d0 c6t5d0 raidz c7t5d0 c6t6d0
c7t6d0 c6t7d0 c7t7d0 c6t8d0 c7t8d0 c6t9d0 c7t9d0 c6t10d0 c7t10d0
# zpool add mypool spare c6t11d0 c7t11d0
# zpool list mypool
NAME    SIZE  USED  AVAIL  CAP  HEALTH  ALTROOT
mypool  2.92T  177K  2.92T   0%  ONLINE  -
```

And we see that we have 2.9 Tb.

### Configuring a Single Array - RAIDZ

Where there is only one array and no possibility of mirroring to storage within the server itself, a RAID approach will probably be required as follows:

```
# zpool create mypool raidz c6t0d0 c6t1d0 c6t2d0 c6t3d0 c6t4d0 \
c6t5d0 c6t6d0 c6t7d0 c6t8d0 c6t9d0 c6t10d0
# zpool add mypool spare c6t11d0
```

This provides us with an 10+1 RAID set and a hotspare:

```
# zpool status mypool
pool: mypool
state: ONLINE
scrub: none requested
config:

    NAME      STATE  READ WRITE CKSUM
    mypool    ONLINE  0   0   0
    raidz1    ONLINE  0   0   0
    c6t0d0    ONLINE  0   0   0
    c6t1d0    ONLINE  0   0   0
    c6t2d0    ONLINE  0   0   0
    c6t3d0    ONLINE  0   0   0
    c6t4d0    ONLINE  0   0   0
    c6t5d0    ONLINE  0   0   0
    c6t6d0    ONLINE  0   0   0
    c6t7d0    ONLINE  0   0   0
    c6t8d0    ONLINE  0   0   0
    c6t9d0    ONLINE  0   0   0
    c6t10d0   ONLINE  0   0   0
    spares
    c6t11d0   AVAIL

errors: No known data errors
```

This gives us about the same amount of space as half our previous mirror as we would expect:

```
# zpool list mypool
NAME    SIZE  USED  AVAIL  CAP  HEALTH  ALTROOT
mypool  1.46T  174K  1.46T   0%  ONLINE  -
```

## Creating Filesystems & Backing them Up

### Creating a File System

In order to store files and directories on the pool we need a file system. This is trivial to create. We can make and mount as many filesystems in the pool as we wish as follows:

```
# zfs create mypool/myfs
```

and see the result of our efforts:

```
# df -h /mypool/myfs
Filesystem      size  used  avail capacity  Mounted on
mypool/myfs    1.3T  32K  1.3T   1%   /mypool/myfs
```

### Backup: Taking a Snapshot

A snapshot is a read-only copy of a file system or volume. It can be created instantaneously, take up almost no room on the storage and can be cloned, backed up, rolled back to and moved from one server to another. They are easy to create:

1. `zfs snapshot mypool/myfs@10-Jun-08`

This gives us a snapshot of our file system called "10-Jun-08". A listing of our file system also shows its snapshot:

```
# zfs list
NAME                                USED AVAIL REFER MOUNTPOINT
mypool                              56.2M 1.30T 34.4K /mypool
mypool/myfs                          56.1M 1.30T 56.1M /mypool/myfs
mypool/myfs@10-Jun-08                 0    - 56.1M -
```

Rolling back to the snapshot (i.e restoring our backup) works as follows:

As an example - first we stupidly delete our data (for our example a copy of /var):

```
# cd /mypool/myfs
# ls
adm  cron  inet  lib  news  opt  sadm  tmp
audit  db  krb5  log  nfs  pkg  saf  yp
cache  fm  ld  lp  nis  preserve  spool
crash  idmap  ldap  mail  ntp  run  svc
# rm -r *
# ls
#
```

Mistake! Then we rollback:

```
# zfs rollback mypool/myfs@10-Jun-08
```

...and the damage is undone:

```
# ls
adm  cron  inet  lib  news  opt  sadm  tmp
audit  db  krb5  log  nfs  pkg  saf  yp
cache  fm  ld  lp  nis  preserve  spool
crash  idmap  ldap  mail  ntp  run  svc
```

## Saving and Sending A Snapshot

We can send the snapshot to another server to create a copy of the file system on that server or simply to store the backups of our file system. First we take our snapshot:

```
$ df -h /mypool/myfs
Filesystem      size  used  avail capacity Mounted on
mypool/myfs    1.3T  48M  1.3T    1%    /mypool/myfs
$ zfs snapshot mypool/myfs@today
```

Then we send it to another server:

```
$ zfs send mypool/myfs@today | ssh otherhost /usr/sbin/zfs receive yourpool/myfs
```

So now we have a snapshot on one server:

```
myhost $ zfs list
NAME                                USED  AVAIL  REFER  MOUNTPOINT
mypool                              53.2M  1.30T  34.4K  /mypool
mypool/myfs                          53.1M  1.30T  53.1M  /mypool/myfs
mypool/myfs@today                    0      -  53.1M  -
```

and a backup copy of it on the other.

```
otherhost $ zfs list
NAME                                USED  AVAIL  REFER  MOUNTPOINT
yourpool                            53.2M  1.30T  34.4K  /yourpool
yourpool/myfs                        53.0M  1.30T  53.0M  /yourpool/myfs
yourpool/myfs@today                  0      -  53.0M  -
```

## Conclusion

In this brief introduction to array administration we have covered the basic tasks involved in setting up file systems on arrays and backing them up by creating snapshots and sending them to another server.

Our hope is that you found it all a lot less painful than you expected.

Naturally there is much more to ZFS administration and the configuration of arrays. The final part of this guide will show you where you can learn more.

## Finding out More

Solaris ZFS manual is at <http://docs.sun.com>

The "man" pages for ZFS: <http://docs.sun.com/app/docs/doc/819-2240/zfs-1m> <http://docs.sun.com/app/docs/doc/819-2240/zpool-1m>

ZFS Learning Centre: [http://www.sun.com/software/solaris/zfs\\_learning\\_center.jsp](http://www.sun.com/software/solaris/zfs_learning_center.jsp)

OpenSolaris ZFS Community: <http://www.opensolaris.org/os/community/zfs/>  
The OpenSolaris ZFS manual will be found here.

ZFS Wiki: <http://www.solarisinternals.com/wiki/index.php?title=Category:ZFS>

OpenSolaris advocacy group presentations: <http://www.opensolaris.org/os/community/advocacy/os-presentations/>  
OpenSolaris mail alias archive: <http://www.opensolaris.org/jive/forum.jspx?forumID=80>

Search for ZFS at <http://blogs.sun.com>

### About the Author

Dominic is a Senior Product Marketing Manager working in Solaris Storage software. He has been at Sun about 10 years working in storage, software and performance. Prior to that he built and led technical teams at Dell and HSBC.

### Acknowledgments

I would like to thank those who took the time to review this article: Ray Dunn, Paul Eggleton, Tim Thomas and Jonathan France.

### Rate this publication

Choices

Your Vote

Great  
Good  
Fair  
Poor



**Business Systems International**

Sun Reseller of the Year & Marketing Award 2009

[www.e-business.com](http://www.e-business.com) | +44 (0)20 7352 7007