Drive Integrity Checks, Archiving

Overview

Working with video creates a tremendous amount of data. In order to avoid ever-increasing costs for fast, redundant, and fully backed up servers, raw footage is archived so that only the project files and the current footage need to be supported (fast/redundant/backed up.)

Background

The primary server has both `AMIFootage` and `TheAttic`. New raw footage is stored in numbered folders on `AMIFootage` and monitored for size. e.g.

```
AMI Footage 1
AMI Footage 2
[...]
AMI Footage 20/
AMI Footage 21/
AMI Footage 22/
[...]
```

As the footage folders are filled to the size of a single hard drive - which varies over time and is not a set size - an archival process occurs. This document covers exactly how to prep the data and drives for archiving, and ignores the standard backup mechanisms in order to keep this document shorter.

Primary server as of 6/1/21 is at IP `10.101.0.4`, and is a container based vm running on amipve2, a Supermicro based 36 bay 4U server. File server name is the same as the old server, `amifs`. Storage for the current live `amifs` server (as of 11/1/21) comes from a bind mount to a zfs pool on the host.

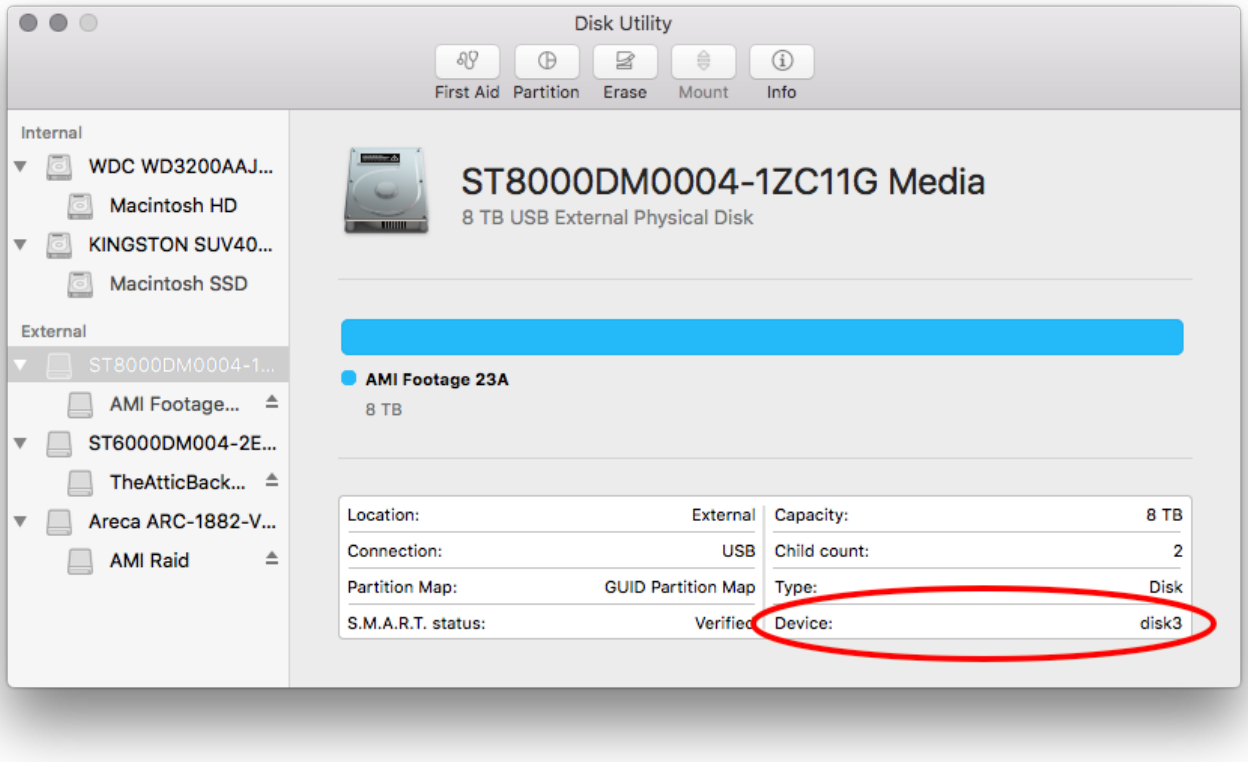
The old server, a Mac Pro with DAS (Direct Attached Storage,) is the backup target, and since it has "toasters" i.e. one or more single or dual hot-pluggable drive bays systems, is the backup target and archive management system.

We could run `badblocks` and `smartctl` natively on amipve2, but any mistakes would be affecting the production system and would be extremely costly, so it's a better idea to use a different system.

Archive Step 1, Drive Identification

Before we run any operations we need to verify we're working with the correct drives.

- Login to the old server. If you know how to use the `screen` binary in ssh that's fine, otherwise please use ARD (Apple Remote Desktop / VNC) and the Terminal application.
- In addition to `Terminal` app, launch `Disk Utility` from the Utilities folder.
- Using `Disk Utility` click on the device (not volume, the volume is the nested item) and look for Device in the lower right hand corner of the window, circled in red here:



- With Terminal.app open, customize this command and paste it in:

```
smartctl -i -d ata /dev/disk3
```

Commonly this will be disk2 or disk3 on this system, disk0 is the boot drive and disk1 is the boot clone. Device numbers cannot be relied on to stay consistent, even if they generally do. Since one of these tests is destructive and has no "Are you sure?" checks, we **must be 100% sure** before running the command.

- Next, paste in this command and verify the model. Example output:

```
amifs:~ admin$ smartctl -i -d ata /dev/disk3
smartctl 7.2 2020-12-30 r5155 [Darwin 15.6.0 x86_64] (local build)
Copyright (C) 2002-20, Bruce Allen, Christian Franke, www.smartmontools.org

=== START OF INFORMATION SECTION ===
Device Model:     ST8000DM0004-1ZC11G
Serial Number:    ZA250LBZ
LU WWN Device Id: 5 000c50 0afd684a4
Firmware Version: DN01
User Capacity:    8,001,563,222,016 bytes [8.00 TB]
Sector Sizes:     512 bytes logical, 4096 bytes physical
Rotation Rate:    7200 rpm
Form Factor:      3.5 inches
Device is:        Not in smartctl database [for details use: -P showall]
ATA Version is:   ACS-3 T13/2161-D revision 5
SATA Version is:  SATA 3.1, 6.0 Gb/s (current: 6.0 Gb/s)
Local Time is:    Mon Nov  1 11:33:48 2021 PDT
SMART support is: Available - device has SMART capability.
SMART support is: Enabled
```

For this system we do **not** want any of the following:

```
Model Family:     Kingston SSDNow UV400/500
Device Model:     KINGSTON SUV400S37240G
Serial Number:    50026B77640B47F1

Model Family:     Western Digital Caviar Blue Serial ATA
Device Model:     WDC WD3200AAJS-41VWA1
Serial Number:    WD-WCARW5800914

Model Family:     Seagate BarraCuda 3.5
Device Model:     ST6000DM004-2EH11C
Serial Number:    ZA1BYVNJ

Read Device Identity failed: empty IDENTIFY data
```

The last entry is for the raid array: naturally we cannot get *disk* info when pointed at a *controller* or the *array* managed by the controller (which itself manages the disks.) Regardless, running badblocks on this device would likely irreversibly destroy all content on the backup aray.

For this example only, we have shown disk3 is a valid target. To avoid mistakes which could involve data loss, I am now switching to a different made-up ID, disk99, for this documentation. Please use the verified disk number you found above, between `Disk Utility` and `smartctl` instead of disk99!

Archive Step 2, Drive Checks

SMART Tests / Drive Self Tests

- In Terminal, use the device identifier verified above in place of `disk99`, and run this command for each drive:

```
smartctl -t long -d ata /dev/disk99
```

You will see text similar to this:

```
smartctl 7.2 2020-12-30 r5155 [Darwin 15.6.0 x86_64] (local build)
Copyright (C) 2002-20, Bruce Allen, Christian Franke, www.smartmontools.org

=== START OF OFFLINE IMMEDIATE AND SELF-TEST SECTION ===
Sending command: "Execute SMART Extended self-test routine immediately in off-line mode".
Drive command "Execute SMART Extended self-test routine immediately in off-line mode"
successful.
Testing has begun.
Please wait 727 minutes for test to complete.
Test will complete after Tue Nov  2 00:54:07 2021 PDT
Use smartctl -X to abort test.
amifs:~ admin$
```

- Note the "test will complete" text. After the noted time, check the status of the extended/long self test:

```
smartctl -a -d ata /dev/disk99
```

You should see "Completed without error" in the `SMART Self-test log` section:

```
SMART Self-test log structure revision number 1
Num  Test_Description    Status                  Remaining  LifeTime(hours)
LBA_of_first_error
# 1  Extended offline    Completed without error       00%         2         -
```

Contact Reinmuth Consulting if you have anything other than "Completed without error" - errors generally mean the drive is unsafe to use, but not always.

Sector Verifications

We now switch to destructive verifications, where every single sector is written to and read from multiple times using known patterns. The intent is to uncover any problems before the drive contains anything valuable.

Do not run badblocks on ANY drive with data!!

- If any volume from the drive is mounted, dismount it. Disk Utility -> select the volume -> Unmount.
- On the Mac Pro backup/archive server, using ARD, with a Terminal window open, run a final verification to ensure you have the right disk number: `smartctl -i -d ata /dev/disk99`.
- This is the start of **destructive** testing. Again, please check and double check disk identifications with expected values. If all looks good and you're positive you have the brand new, completely empty drives identified, proceed with the test:

```
sudo /usr/local/opt/e2fsprogs/sbin/badblocks -b 4096 -wvso
/Users/admin/Documents/badblocks/disk99.txt /dev/disk99
```

^^^ Ensure the above is on one line. It should copy and paste that way, but two lines will not work. ^^^

- You might be prompted for admin's password, enter it and press return if so. No characters will show up as you type. This is normal.
- Output should look similar to:

```
amifs:~ admin$ sudo /usr/local/opt/e2fsprogs/sbin/badblocks -b 4096 -wvso
/Users/admin/Documents/badblocks/disk99.txt /dev/disk99
Password:
Checking for bad blocks in read-write mode
From block 0 to 1953506645
Testing with pattern 0xaa:   0.11% done, 1:36 elapsed. (0/0/0 errors)
```

- This test will take a long time. Typically 1-5 days per drive.
- I recommend opening additional windows so that both of the archive drives can be tested simultaneously. However, the smartctl and badblocks tests cannot be run on the same drive at the same time, they must be run sequentially.
- Do not reboot the machine during the test. All progress will be lost. The drive will not be harmed, however.

- Once the test is complete there is no fanfare, just an empty prompt:

```
amifs:~ admin$
```

Contact Reinmuth Consulting if errors are noted. Check the text file you wrote to roughly here: ~ `/Users/admin/Documents/badblocks/disk99.txt` for content. Healthy drives have files with absolutely nothing in them.

Archive Step 3, Drive Prep

- When both of the previous tests have proven the drives to be healthy, launch Disk Utility and format the drives appropriately. At this time with external HDDs, guidelines are:
  - GUID partition type
  - HFS+ Journaled format
  - Each drive would have the footage folder name and one drive will have A, the other B.
    - e.g. `AMI Footage 23A` and `AMI Footage 23B`

Archive Step 4, Data Transfers

- Launch Carbon Copy Cloner.
- Click on the `Footage Clone, modify as needed` task from the task list on the left.
- Ensure `AMIFootage` is mounted. If not mounted, Finder -> Go -> Connect to Server and paste in:

```
smb://backupserver@10.101.0.4/AMIFootage
```

- Drag the folder you're intending to copy to the Source selection area, and the A drive to the Destination area. Ensure you're copying from the server and not from the local drives.
- In Advanced Settings, ensure the option `Find and replace corrupted files` is set to `Every time the task runs`. This uses checksums to guarantee the data is absolutely identical on the source and destination.
- Create the new AMI Footage ## folder on the server itself, probably from a typical production machine, though this is not strictly necessary. Do not add any content to the folder being copied.
- When the copy is complete, either clone from the server to the other archive drive, or copy from A -> B, modifying the CCC task as needed. Leave the checksumming in place, even though it is slower and takes much longer.

Archive Step 5, Offsite

Please move one of the archive drives offsite, preferably in a shock resistant container with a dessicant to avoid long term moisture damage.

Notes

- Unfortunately `smartctl --scan` shows only internal devices in macOS.
- MacOS does not support SMART over USB, but this company provides a driver to meet out needs: https://binaryfruit.com/drivedx/usb-drive-support

Revision #12
Created 2021-11-01 17:30:05 UTC by Mark Reinmuth
Updated 2021-11-01 20:50:00 UTC by Mark Reinmuth