Falcon Hacking

Chen Xu

<xuchen@brandeis.edu>

$BrandeisEM: ~/emdoc/articles-xml/Flacon-Hack/article.xml 1 2016-01-21 11:03:33 xuchen Exp$

In order to get all the raw frames from FEI Falcon II camera, we have to hack the system. At least, this is the situation up to the time of writing this document. Fortunately, this is a non-destructive hack to FEI camera system, both hardware and software wise. Therefore, we can do whatever we want to do.

I am trying to record all I have done for this project. This doc is written for myself, in case we have to duplicate the system at Janelia Farm of HHMI. If you have interests in doing the similar thing to your Falcon camera, you might also find it is informative and useful. You might not need to ask some very basic and detailed questions to Greg at MRC. Therefore, it might save some Greg's time and energy.

You can also get pdf version of this document here.

[Split HTML / Single HTML]

Table of Contents

1 All the credit goes to Greg McMullan at MRC

I feel extremely lucky for Greg is willing to spend a lot of time to provide instructions, from basic idea to detailed hardware purchase list and software details at computers setup. I follow almost precisely what he instructed, step by step.

All the credit goes to Greg McMullan at MRC. He is the one who comes up this hacking and provided all the information to me.

---

2 The Background

As a CMOS camera, FEI's Falcon camera runs at speed of 17-18 frames per second(fps). However, FEI software doesn't not give all the raw frames. For Falcon I, we simply got a single shot image which is an integration of multiple raw frames. For Falcon II, the Tecnai or Titan software of newer version can maximumly output 7 chunks of frames. Each chunk can be as small as one raw frame, or as large as a sum of a continuous groups of raw frames. Any way, we can only get 7 final chunks (or frames) in the end for each exposure. This might be well likely due to limited computational power of current Windows XP computer that runs the microscope.

There is an 1GbE network connection between Falcon camera controller and Tecnai/Titan computer. The connection between controller to camera head is a pair of fibril optical cables with LC connectors at both ends. This optical cable has speed of 10GbE. All the signals - starting and ending for each exposure and image data itself all go through this pair of fibril optical cables.

The idea is to intercept the signals going through the fibril optical cables and retrieve the image frame information. This is done by using an optical tap, or called optical splitter. The optical tap is a typical network device to allow monitoring the network activities. The monitoring port from this optical tap goes to a special optical network card installed in a powerful Linux computer. Greg managed to modify the network driver, which allows him to retrieve and restore all the raw frames from an exposure.

As you can see, this hacking project involves both hardware and software.

---

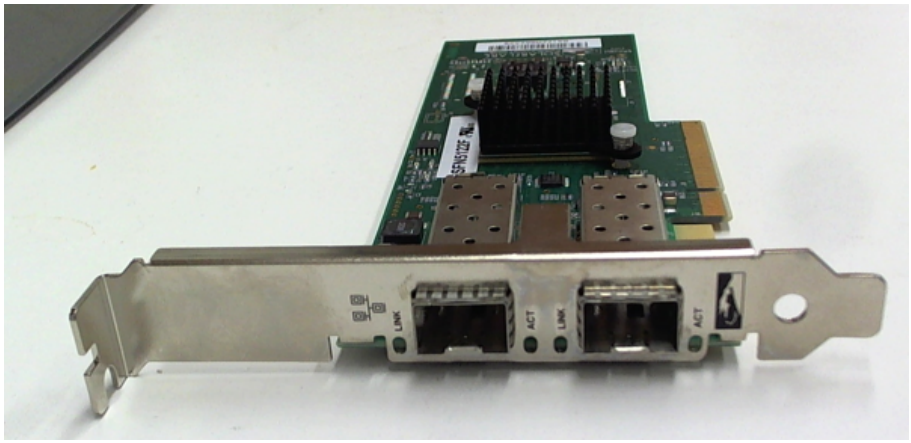
3 Hardware List

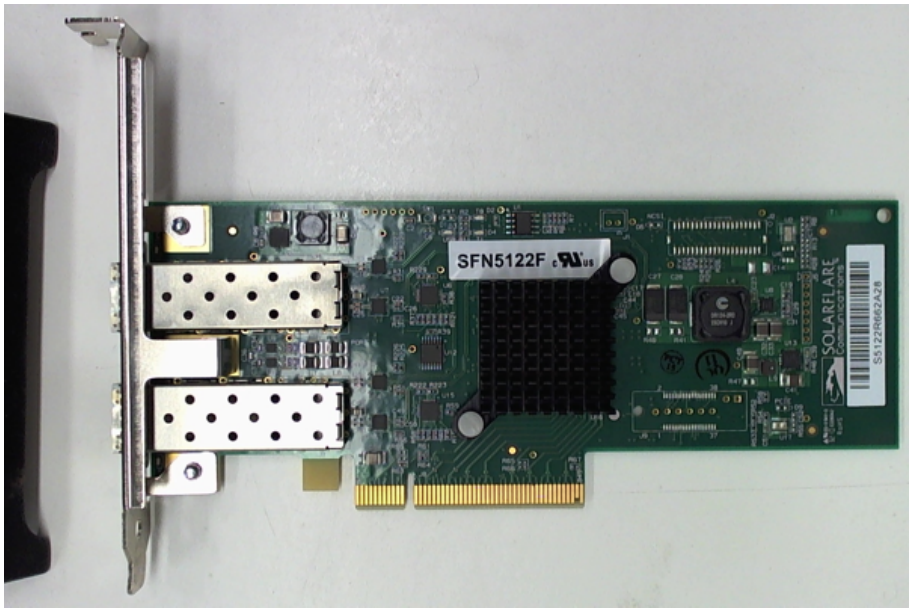
The hardware for this hacking project can be described as three parts: (1) Network Optical Devices, (2) Capture CentOS Linux Computer and (3) FreeNAS Data Storage Tank Computer.

(1) Network Optical Devices:

- 1x SFN5122F Solarflare dual-port 10GbE card.

  This is the main piece of hardware. This is a dual port card. Two image of this card are shown below.





- 2x 10GbE 850nm SFP+ transceivers.

  This works with the optical card. You need two of these, as each of the dual ports on the SolarFlare card will need one. Here are a couple of pictures of it.

- 1x optical tap with LC connectors.

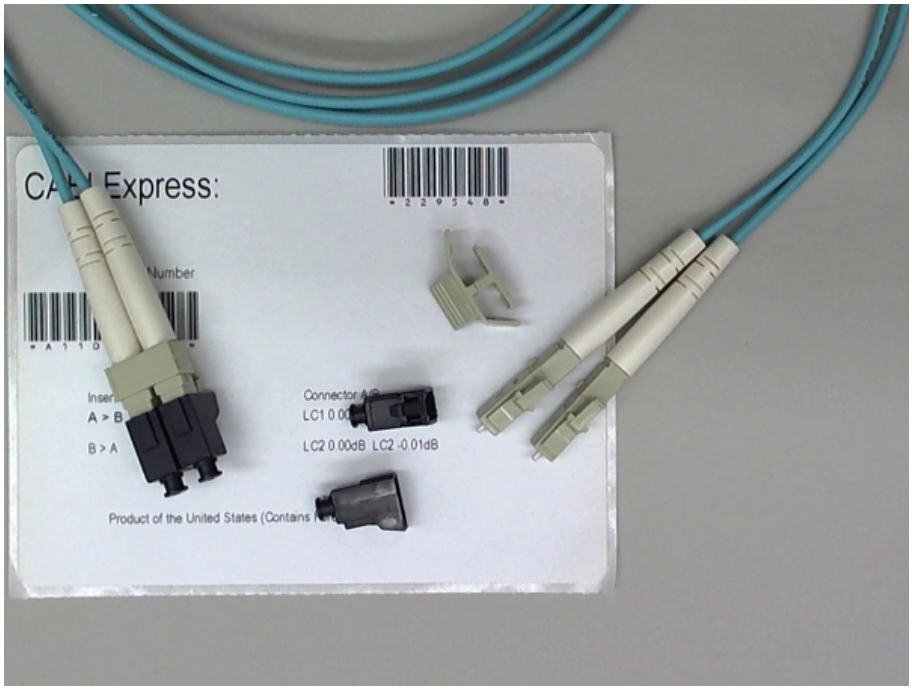
  This is the intercepting unit. As suggested by Greg, we got one from BlackBox. Make sure this is 850 nm, multi mode. Here is the images of this. The part number from BlackBox is TS245A.

- 1x Tap Rackmount Bracket, 19", 1U, from BlackBox. The part Number is TS253.
- 2x LC 50um 850 nm optical cables.

  You can find this kind of optical cable easily. One thing is worth mentioning is that you better off buy the one with "side release clip". One end from monitor port on optical tap needs to be split into two single fibrils so they can be plugged into receiving ports on both of the dual ports on SolarFlare card. With this side release clip, splitting them is much easier. See the second image below.

  The cable I bought from www.cxtec.com has description on the part "229548:2M DUPLEX 50/125 MULTIMODE AQUA OM3 LC/LC RISER JUMPER CABLE WITH SIDE RELEASE CLIPS". You might need to tell them how long you want the cable to be. I bought 2 meters for our setup, you might need longer.

  A couple of images are shown here too.



(2) Capture CentOS Linux Computer

I bought both main computers from iXsystems.com. I have been quite happy with them. One of the reasons I use them is that I know their FreeNAS software system is very robust, easy to configure, has ZFS - one of the most advanced modern filesystems and based on rock solid FreeBSD system for which I am a die-hard fan. The computer case I bought from them for this project is identical to what Gatan uses for their K2 computer.

The main concern is the mount of RAM in the system for this capture computer workstation. We also want to have fast disk array to we can save large images from memory to disk very quickly. Here is what I have:

- Supermicro motherboard X9SRL-F. It has sufficient PCI-E slots: 2 PCI-E 3.0 x8, 2 PCI-E 3.0 x8 (in x16), 2 PCI-E 3.0 x4 (in x8), 1 PCI-E 2.0 x4 (in x8). This is important since we will need PCI-E 3.0 x8 slots for SolarFlare card, another 10GbE network card for local data swapping out and a SATA-3 Host Buss Adapter (HBA) card.
- Intel Quad Core 3.70Ghz Xeon 10MB cache.
- 64GB of RAM.
- 3 500GB SSD disks. I probably should buy 4.
- A Chelsio T420-CR 10 Gigabit Dual Port Card with Twinax cable with SPF+ transceiver modules. This is for quickly swapping out the data saved on SSD to a longer term storage tank.
- A LSI adapter (HBA) [HBA-LSI-9207-8I] for 3 SSD drives. There are only two SATA3 ports on motherboard. SATA2 will slow thing down.

(3) FreeNAS Computer Data Storage Tank

- The some motherboard as Capture Linux Computer, the same CPU, also has 64GB RAM.
- A Chelsio T420-CR 10 Gigabit Dual Port Card.
- 8 4TB SATA3 harddrives.
- A LSI HBA SATA3 adapter for 8 harddrive disks.

---

4 Setup Capture Computer Workstation

The computer shipped to me has been installed with CentOS 6.5 as I requested. The OS sits on a 32GB Solid State Boot Device. Here is the kernel info.

```
# uname -r
    2.6.32-431.el6.x86_64
```

This a standard release kernel. I did not do any update.

I list here some of the steps I performed to this computer.

1. Power up. I see it boots up normally and I get a login prompt. There is no graphic interface of X server installed. So that means I'll have to do every configuration from command line. I feel excited, as it will test how much I still remember this Linux stuff, I am more a FreeBSD person.
2. Install SolarFlare card into PCI-E 3 X8 slot. Reboot. After booting up, I see the card from `dmesg`. And I can see that the Linux RPM driver for this card has also being installed, upon my request to iXsystems.com.

   ```
   # lsmod | grep sfc
   sfc_char               30522  1 onload
   sfc_resource          127811  2 onload,sfc_char
   sfc_affinity           11108  1 sfc_resource
   sfc                   390525  3 onload,sfc_resource,sfc_affinity
   i2c_algo_bit            5935  1 sfc
   mdio                    4769  1 sfc
   sfc_tune               23485  0
   i2c_core               31084  3 sfc,i2c_algo_bit,i2c_i801
   ptp                     9614  2 sfc,e1000e
   ```

   You will see less lines here, as I list here is after I installed the final driver. Initial RPM driver gives less line here.
3. Install this computer box into existing FEI rack in TF30 room. Luckily, there are still two slots to allow me to put both new computers in. I should have ordered the side rack mounting brackets. This is a 4U case, rack mountable. Fortunitely, there are strong supports for them in the rack. They fit in the rack

nicely.

4. Configure 1 1Gb NIC interface and to include it into our existing network of EM suite 192.168.1.0/24. There are 3 ethernet ports on motherboard, 2 from Chelsio card and another 2 from SolarFlare card. It took me some time to figure out which one is which. Once that is known, configuring it becomes easy.

```
# cat /etc/sysconfig/network-scripts/ifcfg-eth2
DEVICE=eth2
HWADDR=0C:C4:7A:00:BD:2E
TYPE=Ethernet
UUID=dc5f71b9-454c-48cc-a00b-7b3c66101bfa
ONBOOT=yes
NM_CONTROLLED=yes
BOOTPROTO=none
IPADDR=192.168.1.20
PREFIX=24
GATEWAY=192.168.1.1
DNS1=129.64.99.205
DNS2=129.64.100.205
DEFROUTE=yes
DOMAIN=brandeis.edu

# ifup eth2

# ifconfig eth2
eth2      Link encap:Ethernet  HWaddr 0C:C4:7A:00:BD:2E
          inet addr:192.168.1.20  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::ec4:7aff:fe00:bd2e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:114424 errors:0 dropped:0 overruns:0 frame:0
          TX packets:26777 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:90517336 (86.3 MiB)  TX bytes:3560653 (3.3 MiB)
          Interrupt:18 Memory:fbd00000-fbd20000
```

5. Install some packages. Later, we need to compile the SolarFlare driver, the *kernel-devel* is the most important one to install.

```
# yum install kernel-devel
```

And I also installed some packages which are needed for full kernel source installation. Some of them might not be neccesary, as complation of SolarFlare driver doesn't really need full kernel source. But there is what I also did:

```
# yum install rpm-build redhat-rpm-config asciidoc hmaccalc perl-ExtUtils-Embed xmlto
# yum install audit-libs-devel binutils-devel elfutils-devel elfutils-libelf-devel
# yum install newt-devel python-devel zlib-devel
```

I also install a couple of packages for my own convenience.

```
# yum -y install man wget
```

I like *tmux* a lot, so also want it. But it is not yet on CentOS yum system. So I have to install the rpm package.

```
# rpm -Uvh http://pkgs.repoforge.org/tmux/tmux-1.6-1.el6.rf.x86_64.rpm
```

6. Now, download *openonload* stable driver and see if we can compile it.

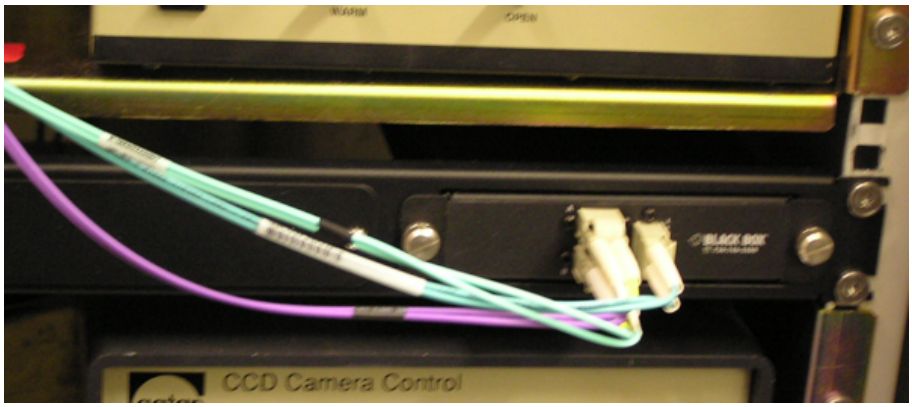
```
# wget http://www.openonload.org/download/openonload-201310-u2.tgz
# tar xvzf openonload-201310-u2.tgz
# cd openonload-201310-u2/scripts
# ./onload_install
```

If that is successful without error message, then it is good. We are ready to move to next step. You might have to install some -libgcc for i686 etc., for 32-bit compatibility. I don't know if it is needed or not, but I included this anyway.

```
# yum -y install glibc-devel.i686
```

7. Install the Rackmount Bracket from BlackBox to the rack.

8. Now install the optical tap to Rackmount Bracket. An image shown its mounting and optical cable connection to the tap is below.



Now lets connect the optical cables. First disconnect the LC connector of the purple color optical cable from FEI Falcon Controller box, and connect it to port A on optical tap. Then use one LC optical cable to connect from Falcon Controller box to port B on the tap. Last, using another LC optical cable, one end goes to port "monitor" on the tap, other end is split to two single fibrils by take them out from the side-release clip. These two fibrils go to receiving port on each of the two ports on the SolarFlare card. The receiving port is the one near "Act" on the card bracket.

As soon as these two fibrils are inserted into receiving ports, the LED on the card light up immediately.

9. It things are setup correctly, Falcon camera should behave just like it always has been. So take an image on microscope to verify it.

10. Now everything looks good, we can bring up the 10GbE Chelsio card ethernet interface. There are two ports on the card, we only need one. Lets find the card and driver info from `dmesg`.

```
# dmesg | grep eth0
        cxgb4 0000:02:00.4: eth0: Chelsio T420-CR rev 2 10GBASE-R SFP+ RNIC PCIe x8 5 GT/s MSI-
        cxgb4 0000:02:00.4: eth0: S/N: PT38130118, E/C: 01234567890123
```

We need to assigned the IP address of this card to a different subnet from 1GbE one. We can give it as 192.168.2.1. After editing the config file, bring it up using "ifup eth0" or reboot.

```
# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
HWADDR=00:07:43:15:36:90
TYPE=Ethernet
UUID=f781d905-1f06-4cbf-b0f7-7fe9a87dd6f2
ONBOOT=yes
NM_CONTROLLED=yes
BOOTPROTO=none
IPADDR=192.168.2.1
PREFIX=24
#GATEWAY=192.168.1.1
#DNS1=129.64.99.205
#DNS2=129.64.100.205
#DEFROUTE=yes
DOMAIN=brandeis.edu
```

This is to connect to FreeNAS file storage tank directly, not for connecting to internet. Therefore, no need to define route and DNS etc.. You can see it from "ifconfig eth0" as below:

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:07:43:15:36:90
          inet addr:192.168.2.1  Bcast:192.168.2.255  Mask:255.255.255.0
          inet6 addr: fe80::207:43ff:fe15:3690/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7956918 errors:0 dropped:0 overruns:0 frame:0
          TX packets:43729928 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3940525722 (3.6 GiB)  TX bytes:65301388658 (60.8 GiB)
          Interrupt:32
```

You can also use "ethtool to see its speed".

```
# ethtool eth0
Settings for eth0:
        Supported ports: [ FIBRE ]
        Supported link modes:   Not reported
        Supported pause frame use: No
        Supports auto-negotiation: No
        Advertised link modes:  Not reported
        Advertised pause frame use: No
        Advertised auto-negotiation: No
        Speed: 10000Mb/s
        Duplex: Full
        Port: Direct Attach Copper
        PHYAD: 0
        Transceiver: internal
        Auto-negotiation: off
        Supports Wake-on: bg
        Wake-on: d
        Current message level: 0x000000ff (255)
                                       drv probe link timer ifdown ifup rx_err tx_err
        Link detected: yes
```

11. It is the time to prepare the disk array. The three SSD drives arrived to have software RAID partition created using GPT. This was upon my request as I want to have them configured as RAID0, but the raid array has not been configured or mounted. We have to do it ourselves.

    When I talked to iXsystems.com and told them that I wanted SSD drives to be configured as RAID0, they told me that RAID0 is not recommended. It is true that RAID0 is less safe for redundancy, but since the images setting on them are only for short period of time before swapping out the larger data tank, I still want RAID0. Considering factors of performance, speed and security, I believe RAID0 is a good choice.

    Since I have to do it from command line, I have to install *mdadm* package first.

    ```
    # yum install mdadm
    ```

    From *dmesg*, I know the three devices related to three SSD driver are ada, adb, adc. So I first created a file `/etc/mdadm.conf` containing following line:

    ```
    DEVICE /dev/sd[abc]
    ARRAY /dev/md0 devices=/dev/sda,/dev/sdb,/dev/sdc
    ```

    Prior to the creation or usage of any RAID devices, the /proc/mdstat file shows no active RAID devices:

    ```
      Personalities :
      read_ahead not set
      Event: 0
      unused devices: none
    ```

    Now, using above configuration and command "mdadm" to create raid0 array:

    ```
    # mdadm -C /dev/md0 --level=raid0 --raid-devices=3 /dev/sda /dev/sdb /dev/sdc
        Continue creating array? yes
        mdadm: array /dev/md0 started.
    ```

    If my SSD has not been prepared with raid partitions, the above command would have done the job. Since I create the new raid devices on pre-existed raid disks, I found that I have to "assemble" them.

    ```
    # mdadm --assemble /dev/md0 /dev/ada /dev/adb /dev/adc
    ```

    And I also need to make new filesystem on the new raid0 array:

    ```
    # mkfs -f ext4 /dev/md0
    ```

    And then I can mount it.

    ```
    # mkdir /mnt/SSD_RAID
    # mount -t ext4 /dev/md0 /mnt/SSD_RAID
    ```

/etc/fstab is needed to update by adding the line, so that the raid array can be mounted when boot.

```
/dev/md0                /mnt/SSD_RAID           ext4    defaults        1 1
```

Now we can see the raid status from /proc/mdstat

```
# cat /proc/mdstat
   Personalities : [raid0]
   md0 : active raid0 sdc[2] sda[0] sdb[1]
         1465159680 blocks super 1.2 512k chunks

         unused devices: none
```

And from command "mdadm --detail /dev/md0":

```
# mdadm --detail /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Sun Feb 23 13:07:23 2014
     Raid Level : raid0
     Array Size : 1465159680 (1397.29 GiB 1500.32 GB)
   Raid Devices : 3
  Total Devices : 3
    Persistence : Superblock is persistent

    Update Time : Sun Feb 23 13:07:23 2014
          State : clean
 Active Devices : 3
Working Devices : 3
 Failed Devices : 0
  Spare Devices : 0

     Chunk Size : 512K

           Name : localhost.localdomain:0  (local to host localhost.localdomain)
           UUID : 555833b1:d9b30cd7:f7e1d2b0:933f23e4
         Events : 0

    Number   Major   Minor   RaidDevice State
       0       8        0        0      active sync   /dev/sda
       1       8       16        1      active sync   /dev/sdb
       2       8       32        2      active sync   /dev/sdc
```

12. Now mount falcon folder on Tecnai computer so capture program can access to the Falcon Gain Reference file and Sensor Defection file. First, at Tecnai computer turn on share for the folder C:\tecnai \data\falcon, and give it share name as Falcon. Then, install CIFS packages on CentOS Linux computer.

    ```
    # yum -y install cifs
    ```

    And we mount it on:

    ```
    # mkdir /mnt/falcon
    # mount -t cifs //192.168.1.2/Falcon -o username=emuser,password="password-for-emuser" /mnt/fal
    ```

    Here, emuser is a dummy userid existed on Tecnai computer.

    We also need to update /etc/fstab to include it

    ```
    //192.168.1.2/Falcon   /mnt/falcon  cifs  username=emuser,password=passwd-for-emuser 0 0
    ```

13. We need to prepare the memory allocation for capture program. Currently, on our system we have following three lines in `/etc/sysctl.conf`.

    ```
    vm.nr_hugepages = 6144
    kernel.shmmax = 38927335424
    kernel.shmall = 38927335424
    ```

    After modifying `/etc/sysctl.conf`, the computer needs to reboot to take the change into effect.

---

5 Setup FreeNAS Data Tank Computer

The computer shipped with FreeNAS 9.2.1 installed. FreeNAS is based on open source FreeBSD system and can be configured everything from a web browser. It is a polished product. If you know FreeBSD system, then managing FreeNAS system is very simple. FreeNAS system also have wonderful documentation for you to check.

1. Connect a monitor and keyboard, power up the computer.
2. After booting finished, select option 1 - configure the network interface. I only configure `em0` at this point. And I give it IPaddress as 192.168.1.21.
3. After the network interface `em0` is configured, I can open a web browser from any computer in the same LAN and point to http://192.168.1.21 .
4. As I read from FreeNAS release new, the recent release 9.2.1 has a few bugs related to jail and cifs. FreeNAS release team recommended to upgrade to a latest release 9.2.1.1. I did that. The upgrading process via GUI is very simple and quick. Within a few minutes, my system was upgraded to the latest 9.2.1.1.
5. Configure the other 10GbE network interface `cxgb0`. I give it IP address 192.168.2.2. After this setup, I can ping this address from Capture Linux Computer. So I know the 10GbE connection is established between the two Chelsio cards in two computers.
6. Configure storage risk array. I have 8 4TB WD disks. I go ahead to configure them at RAIDZ, even the system says it is not standard and recommend RAIDZ2. Again, because the data on this array is not critically needed for redundancy and RAIDZ has better speed performance, I still go with RAIDZ. After it configured, I have ~23TB storage space, available with two network interfaces, 1GbE and 10GbE. This part of configuration process is so simple that I really love it.
7. I need to export large data tank disk array to Capture Linux Computer, so the copying data between SSD and this data storage tank is easy.

   Configure NFS service for this RAIDZ array. I export it to the client IP address of 192.168.2.1 which is the 10GbE Chelsio network interface on Capture Linux Computer. I define -maproot = root as export option. After saving the settings, the system is ready to be mounted from Capture Linux Computer.

---

6 Final System Setup

We are at step to put everything together.

1. Ask Greg for his modified version of *openonload* driver and install it. Like before, as `root`, do the following.

   ```
   # mkdir greg
   # cd greg
   # tar xvf modified_oo.tar
   # cd openonload-201310-u2/scripts
   # ./onload_install
   ```

   In the case the onload driver has been already installed, one might have to uninstall it first.

   ```
   # onload_uninstall
   # ./onload_install
   ```

2. On Capture CentOS Linux computer, create a user with username as `capture`. The UID and GID both equal to 29998. And also make sure the user `capture` can do "sudo" so it should belong to sudoers.
3. login to Capture CentOS Linux computer as `capture`. Ask Greg for his *capture* program and all the header files. Place this file `capture.tar.bz2` in capture's home directory.

   ```
   % tar xjvf capture.tar.bz2
   % cd CAPTURE
   % ls -l
   total 12
   drwxr-xr-x. 2 capture capture 4096 Feb 25 07:15 lib
   drwxr-xr-x. 2 capture capture 4096 Feb 25 13:03 scripts
   drwxr-xr-x. 2 capture capture 4096 Feb 25 07:16 src
   % cd lib
   % make clean && make
   ```

```
% cd ../src
% make clean && make
```

If things go correctly, you will have a compiled program capture in src directory. You might have to install packages *expat* and *expat-devel*.

4. Now it is the moment. Make a directory from SSD_RAID and from it run:

```
% sudo ifconfig eth4 mtu 9000
% sudo ifconfig eth5 mtu 9000
% ~capture/CAPTURE/src/capture
```

The program will spit out some monitoring info like this:

```
20_17_34:Idle     :370, 0:796, 72650:: 2100 76 76: Range 10000 0
```

Pay attention to the value in the place of :796, 72650:. On our system it is :330, 73444:. If you see the two numbers the other way around, swap the two fibrils to SolarFlare card plug position. We have the following spit out lines from our system.

```
OUTPUT USERID (-u) 29998
OUTPUT grouid (-g) 29998
Minimum movie size to be saved  (-m) 1.000000 GB
NOTE: Saving only movies with more than 16 frames
NY = 4096 row_start 0 row_end 4096 nx 4096 g_lda 4096
Reading Defect file /mnt/falcon/SensorDefects.xml
XML start defects
XML start row
XML start col
XML start col
XML start point
NUMBER of DEFECTS 4
TEST g_lda 4096 ny 4096 nx 4096
0 : ROW defect 446 : 0
1 : Column defect 1199
2 : Column defect 1815
3 : POINT defect 2987 347 2
RETURNING 4
Read defect file: /mnt/falcon/SensorDefects.xml  with 4 defects
Correcting for 4 defects
CLOSED FILE
Gain 609580.732702 (27330.847552) defined using 13649329 points
NEW MEAN (excluding defects) 1.000320 dmin 42.203042 dmax 0.818717 IC 16764929
Gain file read and gain correction stored as gain_2014_02_27-11_48_40.mrc
Trying to initialize memory for 160 frames
Initializing enough memory for 160 frames
ATTEMPTING to allocate 5121 huge pages for buffer of size  10741612544 bytes 10 GB
FRAME SIZE 67108864
SHMID1 0
NOW MEMSET 10741612544
AFTER MEMSET
Allocate memory for gain and dark reference field
MAX number of interfaces 2
Alloca interface eth4
Alloca interface eth5
Global->threads_n 2
THREAD id 1 ->vis_n 1
THREAD id 0 ->vis_n 1
>>>>>>>>>>>>>>>> SET FILTERS 0
>>>>>>>>>>>>>>>> SET FILTERS 1
Unknown event
Unknown event
Unknown event
Unknown event
11_49_41:Idle     :330, 0:330, 73446:: 0 0 0: Range 10000 0
11_50_41:Idle     :330, 0:330, 73446:: 0 0 0: Range 10000 0
11_51_41:Idle     :330, 0:330, 73446:: 0 0 0: Range 10000 0
11_52_41:Idle     :330, 0:330, 73446:: 0 0 0: Range 10000 0
11_53_41:Idle     :330, 0:330, 73446:: 0 0 0: Range 10000 0
11_54_41:Idle     :330, 0:330, 73446:: 0 0 0: Range 10000 0
11_55_41:Idle     :329, 0:329, 73446:: 0 0 0: Range 10000 0
11_56_41:Idle     :330, 0:330, 73446:: 0 0 0: Range 10000 0
11_57_41:Idle     :329, 0:329, 73446:: 0 0 0: Range 10000 0
11_58_41:Idle     :330, 0:330, 73446:: 0 0 0: Range 10000 0
11_59_41:Idle     :330, 0:329, 73446:: 0 0 0: Range 10000 0
12_00_41:Idle     :330, 0:330, 73446:: 0 0 0: Range 10000 0
12_01_41:Idle     :330, 0:330, 73446:: 0 0 0: Range 10000 0
12_02_41:Idle     :330, 0:330, 73446:: 0 0 0: Range 10000 0
```

5. If you take a single shot Falcon image, you should see MRC files in the directory!
