http://www.flickr.com/photos/dtrimarchi/6815004394/

# OST data migrations using ZFS snapshot/send/receive

**Tom Crowe**
Research Technologies
High Performance File Systems
hpfs-admin@iu.edu
Indiana University

# Abstract

Data migrations can be time consuming and tedious, often requiring large maintenance windows of downtime. Some common reasons for data migrations include aging and/or failing hardware, increases in capacity, and greater performance. Traditional file and block based “copy tools” each have pros and cons, but the time to complete the migration is often the core issue. Some file based tools are feature rich, allowing quick comparisons of date/time stamps, or changed blocks inside a file. However examining multiple millions, or even billions of files takes time. Even when there is little no no data churn, a final "sync" may take hours, even days to complete, with little data movement. Block based tools have fairly predictable transfer speeds when the block device is otherwise "idle", however many block based tools do not allow "delta" transfers. The entire block device needs to be read, and then written out to another block device to complete the migration

# Abstract - continued

ZFS backed OST’s can be migrated to new hardware or to existing reconfigured hardware, by leveraging ZFS snapshots and ZFS send/receive operations. The ZFS snapshot/send/receive migration method leverages incremental data transfers, allowing an initial data copy to be "caught up" with subsequent incremental changes. This migration method preserves all the ZFS Lustre properities (mgsnode, fsname, network, index, etc), but allows the underlying zpool geometry to be changed on the destination. The rolling ZFS snapshot/send/receive operations can be maintained on a per OST basis, allowing granular migrations.

# Abstract - continued

This migration method greatly reduces the final "sync" downtime, as rolling snapshot/send/receive operations can be continuously run, thereby pairing down the delta's to the smallest possible amount.  There is no overhead to examine all the changed data, as the snapshot "is" the changed data. Additionally, the final sync can be estimated from previous snapshot/send/receive operations, which supports a more accurate downtime window.

This presentation will overview how Indiana University is leveraging ZFS snapshots and ZFS send/receive to migrate OST data.

# Abstract - highlights

- ZFS snap/send/receive is another tool in the tool box: not right for every job
- ZFS snap/send/receive can reduce “final sync” downtime(s)
- OST/Lustre data/structure remains the same; ZFS data is rebuilt
- Migrate from older hardware to newer hardware
- Change zpool geometry (via migration)

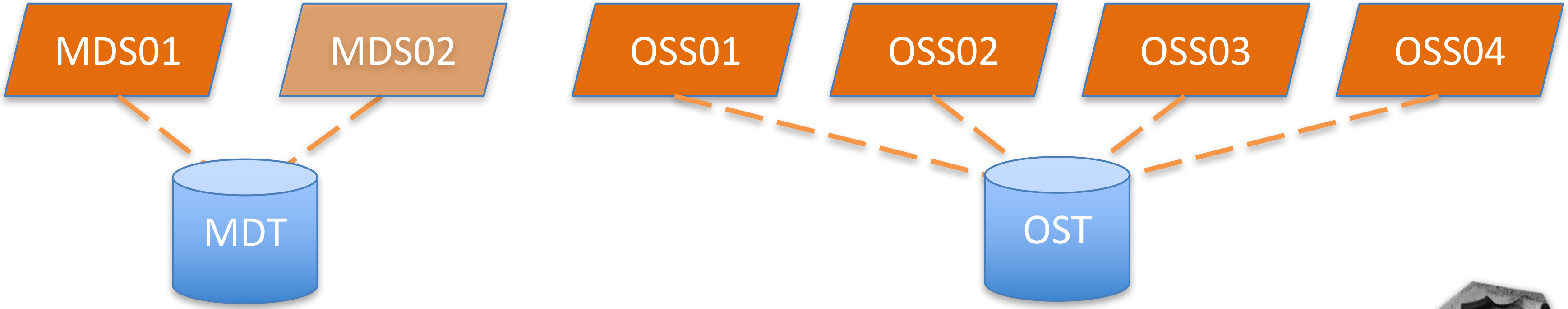
# Reasoning

- Migrated with file based tools, millions of tiny files, very old hardware
- Months of effort
- Final sync/cutover was still very long with little data movement
- ZFS was the intended target
- Looking for a “better way” the next time

# Production Environment Overview

(2) MDS (4) OSS

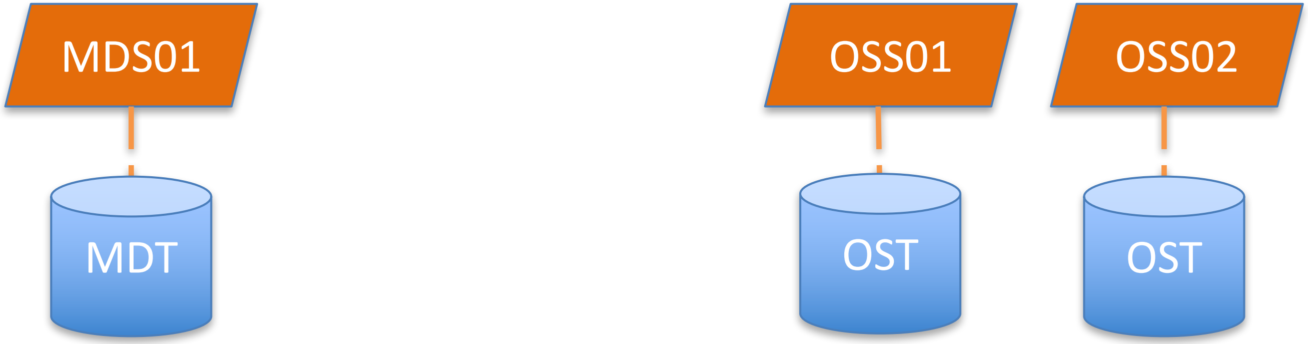
- MDS active/passive for manual failover
- OSS active/active w/manual failover to any node
- Bare Metal HW
- Centos 6.8, Lustre 2.8.0, ZFS 0.6.5.2-1
- MDT and OST's are HW raid



# Demo Environment Overview

(1) MDS (2) OSS

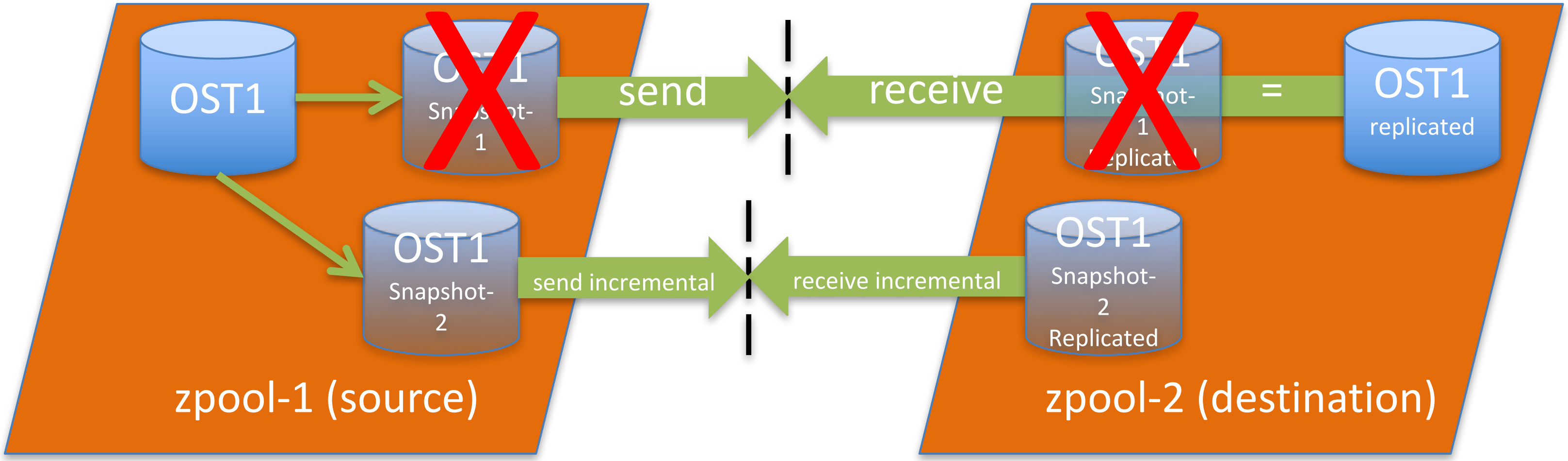
- Built from same XCAT postscripts as production
- KVM Guests (virtual)
- Centos 6.8, Lustre 2.8.0, ZFS 0.6.5.2-1
- MDT and OST's are ZFS raid



# Basic OST Migration

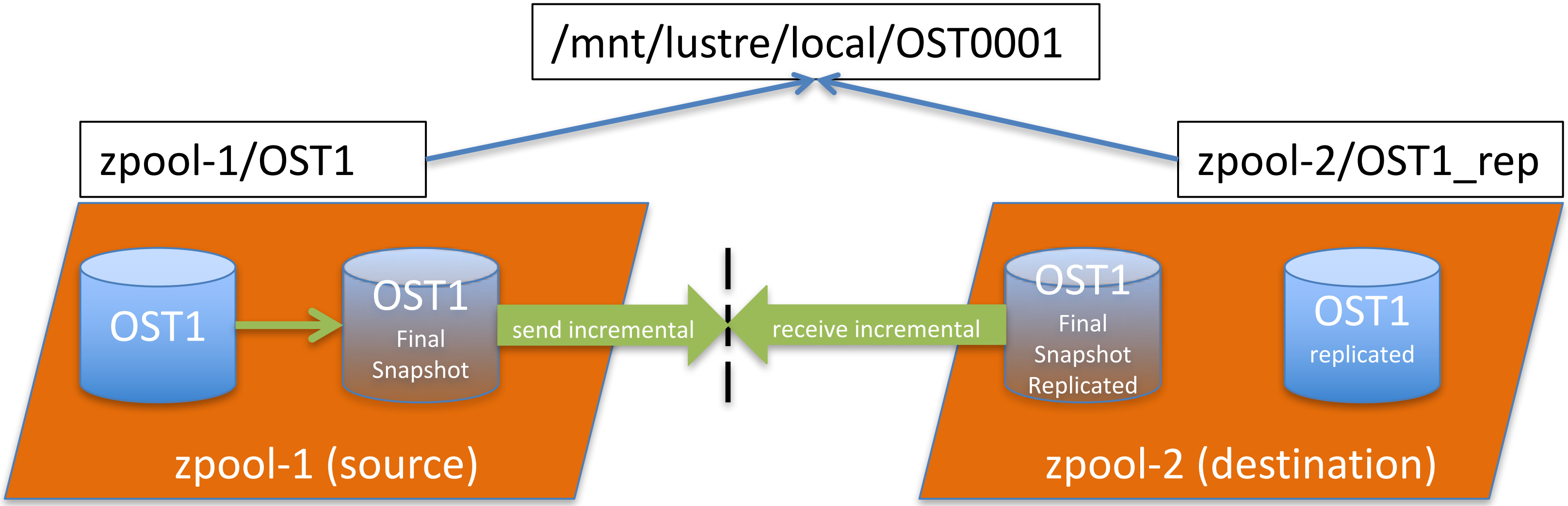
# Basic OST Migration

1. Take a snapshot of source OST
2. Transport the OST snapshot (initial full copy) via zfs send/receive
3. Repeat snapshot/send/receive (incremental)
4. Clean up old snapshots along the way



# Basic OST Migration - continued

5. Stop Lustre (unmount OST)
6. Final snapshot/send/receive
7. Edit /etc/ldev.conf, start “new” device (mount)



# Basic OST Migration – example

Freshly formatted demo file system, two OSTs, no data yet

```
[root@demo_oss01 ~]# df -hP
Filesystem                   Size  Used Avail Use% Mounted on
/dev/mapper/vg00-lv01         16G  4.4G   11G  30% /
tmpfs                        939M     0  939M   0% /dev/shm
/dev/sda1                    488M   84M  379M  19% /boot
/dev/sda2                    488M  396K  462M   1% /boot-rcvy
/dev/mapper/vg00-lv00        976M  1.3M  924M   1% /rcvy
/dev/mapper/vg00-lv05        2.0G   11M  1.8G   1% /scratch
/dev/mapper/vg00-lv03        976M  199M  727M  22% /var
/dev/mapper/vg00-lv02        976M  1.3M  924M   1% /var-rcvy
osspool-01/ost-demo17-0000   3.8G  2.2M  3.8G   1% /mnt/lustre/local/demo-OST0000
osspool-02/ost-demo17-0001   3.9G  2.0M  3.9G   1% /mnt/lustre/local/demo-OST0001
```

# Basic OST Migration – example

Minimum inodes.

```
[root@demo_oss01 ~]# df -i
Filesystem            Inodes IUsed  IFree IUse% Mounted on
/dev/mapper/vg00-lv01
                     1048576 72178 976398    7% /
tmpfs                 240240     1 240239    1% /dev/shm
/dev/sda1              32768    46  32722    1% /boot
/dev/sda2              32768    11  32757    1% /boot-rcvy
/dev/mapper/vg00-lv00  65536    11  65525    1% /rcvy
/dev/mapper/vg00-lv05 131072    13 131059    1% /scratch
/dev/mapper/vg00-lv03  65536  1540  63996    3% /var
/dev/mapper/vg00-lv02  65536    11  65525    1% /var-rcvy
osspool-01/ost-demo17-0000
                      139995   223 139772    1% /mnt/lustre/local/demo-OST0000
osspool-02/ost-demo17-0001
                      141747   223 141524    1% /mnt/lustre/local/demo-OST0001
```

# Basic OST Migration – example

Change record size in zfs dataset from default 128k to 1k, so we can squeeze lots of little files.

```
[root@demo_oss01 ~]# zfs get recordsize osspool-01/ost-demo17-0000
NAME                        PROPERTY    VALUE    SOURCE
osspool-01/ost-demo17-0000  recordsize  128K     default

[root@demo_oss01 ~]# zfs set recordsize=1k osspool-01/ost-demo17-0000
[root@demo_oss01 ~]# zfs set recordsize=1k osspool-02/ost-demo17-0001

[root@demo_oss01 ~]# zfs get recordsize osspool-01/ost-demo17-0000
NAME                        PROPERTY    VALUE    SOURCE
osspool-01/ost-demo17-0000  recordsize  1K       local
```

# Basic OST Migration – example

After creating a million tiny files (less than 1kb each)

```
[root@demo_client ~]# lfs df -h ; lfs df -i
UUID                       bytes        Used   Available Use% Mounted on
demo17-MDT0000_UUID         2.8G        1.3G        1.5G  46% /mnt/demo17[MDT:0]
demo17-OST0000_UUID         3.8G        1.7G        2.0G  47% /mnt/demo17[OST:0]
demo17-OST0001_UUID         3.8G        1.9M        3.8G   0% /mnt/demo17[OST:1]

filesystem summary:         7.6G        1.7G        5.8G  23% /mnt/demo17

UUID                      Inodes       IUsed       IFree IUse% Mounted on
demo17-MDT0000_UUID      1428288     1025216      403072  72% /mnt/demo17[MDT:0]
demo17-OST0000_UUID      3160758     1034687     2126071  33% /mnt/demo17[OST:0]
demo17-OST0001_UUID      3998782         223     3998559   0% /mnt/demo17[OST:1]

filesystem summary:      1428288     1025216      403072  72% /mnt/demo17
```

# Basic OST Migration – example

Check destination zpool has enough free space (source was 1.9GB)

```
[root@demo_oss01 ~]# zfs list
NAME                          USED  AVAIL  REFER  MOUNTPOINT
osspool-01                   1.75G  2.04G  27.2K  /osspool-01
osspool-01/ost-demo17-0000   1.75G  2.04G  1.75G  /osspool-01/ost-demo17-0000
osspool-02                   1.99M  3.83G  24.0K  /osspool-02
osspool-02/ost-demo17-0001   1.89M  3.83G  1.89M  /osspool-02/ost-demo17-0001

[root@demo_oss01 ~]# zpool list
NAME         SIZE  ALLOC   FREE  EXPANDSZ   FRAG    CAP  DEDUP  HEALTH  ALTROOT
osspool-01  4.91G  2.20G  2.71G         -    76%    44%  1.00x  ONLINE  -
osspool-02  5.94G  3.01M  5.93G         -     0%     0%  1.00x  ONLINE  -
```

# Basic OST Migration – example

Check for existing snapshots. Take a snapshot. Use a meaningful snapname (date/time)

```
[root@demo_oss01 ~]# zfs list -t snap
no datasets available


[root@demo_oss01 ~]# zfs snap osspool-01/ost-demo17-0000@`date +%Y%m%d-%H%M%S`


[root@demo_oss01 ~]# zfs list -t snap
NAME                                         USED  AVAIL  REFER  MOUNTPOINT
osspool-01/ost-demo17-0000@20170509-102649      0      -  1.75G  -
```

# Basic OST Migration – example

Initial zfs send is a “full” copy. Use –R (replication stream) in zfs send. DON’T SEND TO SAME POOL

```
[root@demo_oss01 ~]# zfs send -Rv osspool-01/ost-demo17-0000@20170509-102649 | zfs
receive osspool-02/ost-demo17-0000_replicated
send from @ to osspool-01/ost-demo17-0000@20170509-102649 estimated size is 1.20G
total estimated size is 1.20G
TIME        SENT   SNAPSHOT
10:29:00    149M   osspool-01/ost-demo17-0000@20170509-102649
10:29:01    206M   osspool-01/ost-demo17-0000@20170509-102649
10:29:02    247M   osspool-01/ost-demo17-0000@20170509-102649
...
11:03:34   1.99G   osspool-01/ost-demo17-0000@20170509-102649
11:03:35   1.99G   osspool-01/ost-demo17-0000@20170509-102649
11:03:37   1.99G   osspool-01/ost-demo17-0000@20170509-102649
[root@demo_oss01 ~]#
```

# Basic OST Migration – example

Results following initial snap/send/receive

```
[root@demo_oss01 ~]# zfs list -t snap
NAME                                              USED   AVAIL   REFER   MOUNTPOINT
osspool-01/ost-demo17-0000@20170509-102649        620K       -   1.75G   -
osspool-02/ost-demo17-0000_replicated@20170509-102649     0       -   1.48G   -

[root@demo_oss01 ~]# zfs list
NAME                                    USED   AVAIL   REFER   MOUNTPOINT
osspool-01                             1.77G   2.03G   27.2K   /osspool-01
osspool-01/ost-demo17-0000             1.76G   2.03G   1.76G   /osspool-01/ost-demo17-0000
osspool-02                             3.00G    846M   24.0K   /osspool-02
osspool-02/ost-demo17-0000_replicated  1.48G    846M   1.48G   /osspool-02/ost-demo17-
0000_replicated
osspool-02/ost-demo17-0001             1.52G    846M   1.52G   /osspool-02/ost-demo17-0001
```

# Basic OST Migration – example

Take a subsequent snapshot. Use a meaningful snapname (date/time)

```
[root@demo_oss01 ~]# zfs snap osspool-01/ost-demo17-0000@`date +%Y%m%d-%H%M%S`

[root@demo_oss01 ~]# zfs list -t snap
NAME                                                  USED  AVAIL  REFER  MOUNTPOINT
osspool-01/ost-demo17-0000@20170509-102649            620K      -  1.75G  -
osspool-01/ost-demo17-0000@20170509-130143               0      -  1.76G  -
osspool-02/ost-demo17-0000_replicated@20170509-102649    0      -  1.48G  -
```

# Basic OST Migration – example

Subsequent zfs send is a “incremental” (only changes between snapshots sent). Use –R and –i for incremental

```
[root@demo_oss01 ~]#zfs send -Rv -i osspool-01/ost-demo17-0000@20170509-102649 osspool-
01/ost-demo17-0000@20170509-130143 | zfs receive osspool-02/ost-demo17-0000_replicated
send from @20170509-102649 to osspool-01/ost-demo17-0000@20170509-130143 estimated size
is 8.72M
total estimated size is 8.72M
TIME        SENT   SNAPSHOT
13:09:36    322K   osspool-01/ost-demo17-0000@20170509-130143
```

# Basic OST Migration – example

Remove old snapshots, review current snapshot

```
[root@demo_oss01 ~]# zfs destroy osspool-01/ost-demo17-0000@20170509-102649
(DOES NOT CONFIRM, IT JUST DELETES IT)
[root@demo_oss01 ~]# zfs destroy osspool-02/ost-demo17-0000_replicated@20170509-102649

[root@demo_oss01 ~]# zfs list -t snap
[root@demo_oss01 lustre]# zfs list -t snap
NAME                                                 USED  AVAIL  REFER  MOUNTPOINT
osspool-01/ost-demo17-0000@20170509-130143           255M      -  1.76G  -
osspool-02/ost-demo17-0000_replicated@20170509-130143   0      -  1.49G  -
```

# Basic OST Migration – example

Stop the OST, import zpool (if necessary), create final snapshot

```
[root@demo_oss01 ~]# grep OST0000 /etc/ldev.conf
demo_oss01     -       demo-OST0000           zfs:osspool-02/ost-demo17-0000

[root@demo_oss01 ~]# service lustre stop demo-OST0000
Unmounting /mnt/lustre/local/demo-OST0000

[root@demo_oss01 ~]# zpool import osspool-01

[root@demo_oss01 ~]# zfs snap osspool-01/ost-demo17-0000@`date +%Y%m%d-%H%M%S`
[root@demo_oss01 ~]# zfs list -t snap
NAME                                            USED  AVAIL  REFER  MOUNTPOINT
osspool-01/ost-demo17-0000@20170509-130143      270M      -  1.76G  -
osspool-01/ost-demo17-0000@20170510-094747         0      -  1.77G  -
osspool-02/ost-demo17-0000_replicated@20170509-130143  0  -  1.49G  -
```

# Basic OST Migration – example

Subsequent zfs send is a “incremental” (only changes between snapshots sent). Use –R and –I for incremental

```
[root@demo_oss01 ~]# zfs send -Rv -i osspool-01/ost-demo17-0000@20170509-130143
osspool-01/ost-demo17-0000@20170510-094747 | zfs receive osspool-02/ost-demo17-
0000_replicated
```

```
send from @20170509-130143 to osspool-01/ost-demo17-0000@20170510-094747 estimated size
is 192M
total estimated size is 192M
TIME          SENT    SNAPSHOT
09:50:42     1.61M    osspool-01/ost-demo17-0000@20170510-094747
09:50:43     48.7M    osspool-01/ost-demo17-0000@20170510-094747
09:50:44     53.6M    osspool-01/ost-demo17-0000@20170510-094747
...
09:52:51      381M    osspool-01/ost-demo17-0000@20170510-094747
09:52:52      382M    osspool-01/ost-demo17-0000@20170510-094747
09:52:53      383M    osspool-01/ost-demo17-0000@20170510-094747
```

# Basic OST Migration – example

Modify /etc/ldev.conf to reflect “new” OST, start OST

```
[root@demo_oss01 ~]# grep OST0000 /etc/ldev.conf
demo_oss01      -       demo-OST0000            zfs:osspool-02/ost-demo17-
0000_replicated

[root@demo_oss01 ~]# service lustre start demo-OST0000
Mounting osspool-02/ost-demo17-0000_replicated on /mnt/lustre/local/demo-OST0000
```

# Basic OST Migration – review

1. identify sources, destination and existing snaps

```
zfs list ; zpool list ; zfs list –t snap
```

2. take a snapshot

```
zfs snap src_zpool/vdev@snapname1
```

3. transport “full” snapshot

```
zfs send -Rv src_zpool/vdev@snapname1 | zfs receive dst_zpool/vdev_rep
```

4. take another snapshot

```
zfs snap src_zpool/vdev@snapname2
```

5. transport “incremental” snapshot

```
zfs send -Rv -i src_zpool/vdev@snapname1 src_zpool/vdev@snapname2 | zfs
receive dst_zpool/vdev_rep
```

6. clean up old snapshots

```
zfs destroy src_zpool/vdev@snapname1
```

# Basic OST Migration – review

7. Repeat steps 4, 5 and 6 as long as needed until “final sync”
8. Stop Lustre
   - `service lustre stop`
9. repeat steps 4 and 5 (snap, send/receive)
10. Modify/update /etc/ldev.conf with new device
11. Start Lustre
    - `service lustre start`

Don‘t forget to cleanup/remove old snaps and source OST

# Basic OST Migration – Summary

- The rhythm is snap, send/receive, destroy until the “final sync”.
- For final sync, stop Lustre/OST, then snap, send/receive, start Lustre/OST
- There are performance and capacity impacts; its not magic.
  - Zpool capacity needs to be monitored
  - IO overhead for ZFS snap/send/receive and destroy
  - Zpool iostat can monitor activity
- Efficient transport of incremental changes
- This is appropriate for:
  - “controlled” downtime
  - Back end storage changes (capacity, age, speed)
  - Zpool geometry changes

# Basic OST Migration – ZFS vs rsync

comparison on demo environment

Initial full ZFS snap/send/receive takes ~34 minutes

```
TIME          SENT    SNAPSHOT
10:29:00      149M    osspool-01/ost-demo17-0000@20170509-102649
10:29:01      206M    osspool-01/ost-demo17-0000@20170509-102649
10:29:02      247M    osspool-01/ost-demo17-0000@20170509-102649
...
11:03:34     1.99G    osspool-01/ost-demo17-0000@20170509-102649
11:03:35     1.99G    osspool-01/ost-demo17-0000@20170509-102649
11:03:37     1.99G    osspool-01/ost-demo17-0000@20170509-102649
```

# Basic OST Migration – ZFS vs rsync

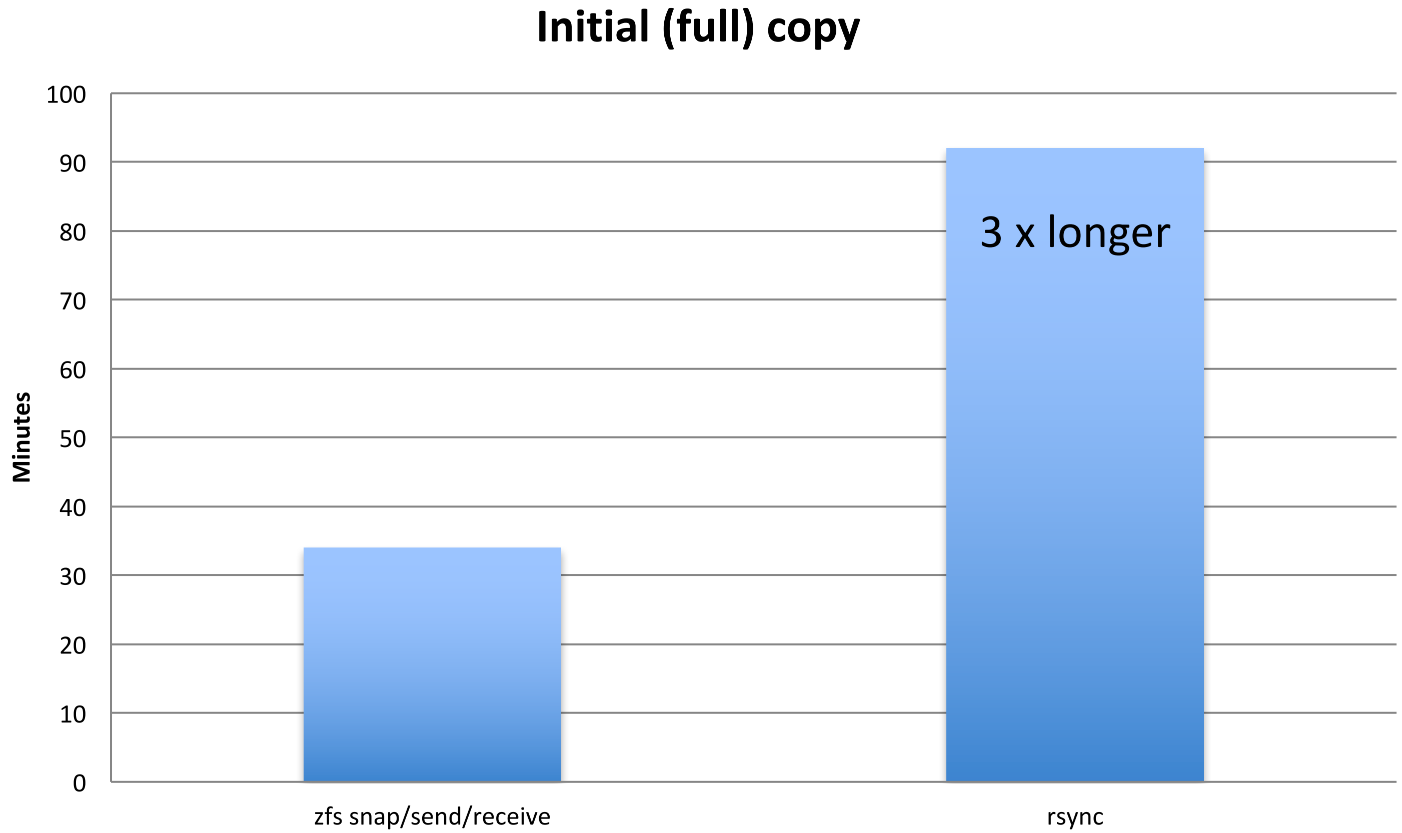
comparison on demo environment

Initial full rsync of the same data takes ~92 minutes

```
rsync -azh --no-whole-file . ${DEST}/
```

# Basic OST Migration – ZFS vs rsync

comparison on demo environment



# Basic OST Migration – ZFS vs rsync

## comparison on demo environment

Subsequent incremental ZFS snap/send/receive takes ~2 minutes

~150,000 files were appended to (8 bytes)

```
TIME         SENT    SNAPSHOT
09:50:42    1.61M    osspool-01/ost-demo17-0000@20170510-094747
09:50:43    48.7M    osspool-01/ost-demo17-0000@20170510-094747
09:50:44    53.6M    osspool-01/ost-demo17-0000@20170510-094747
...
09:52:51     381M    osspool-01/ost-demo17-0000@20170510-094747
09:52:52     382M    osspool-01/ost-demo17-0000@20170510-094747
09:52:53     383M    osspool-01/ost-demo17-0000@20170510-094747
```

# Basic OST Migration – ZFS vs rsync

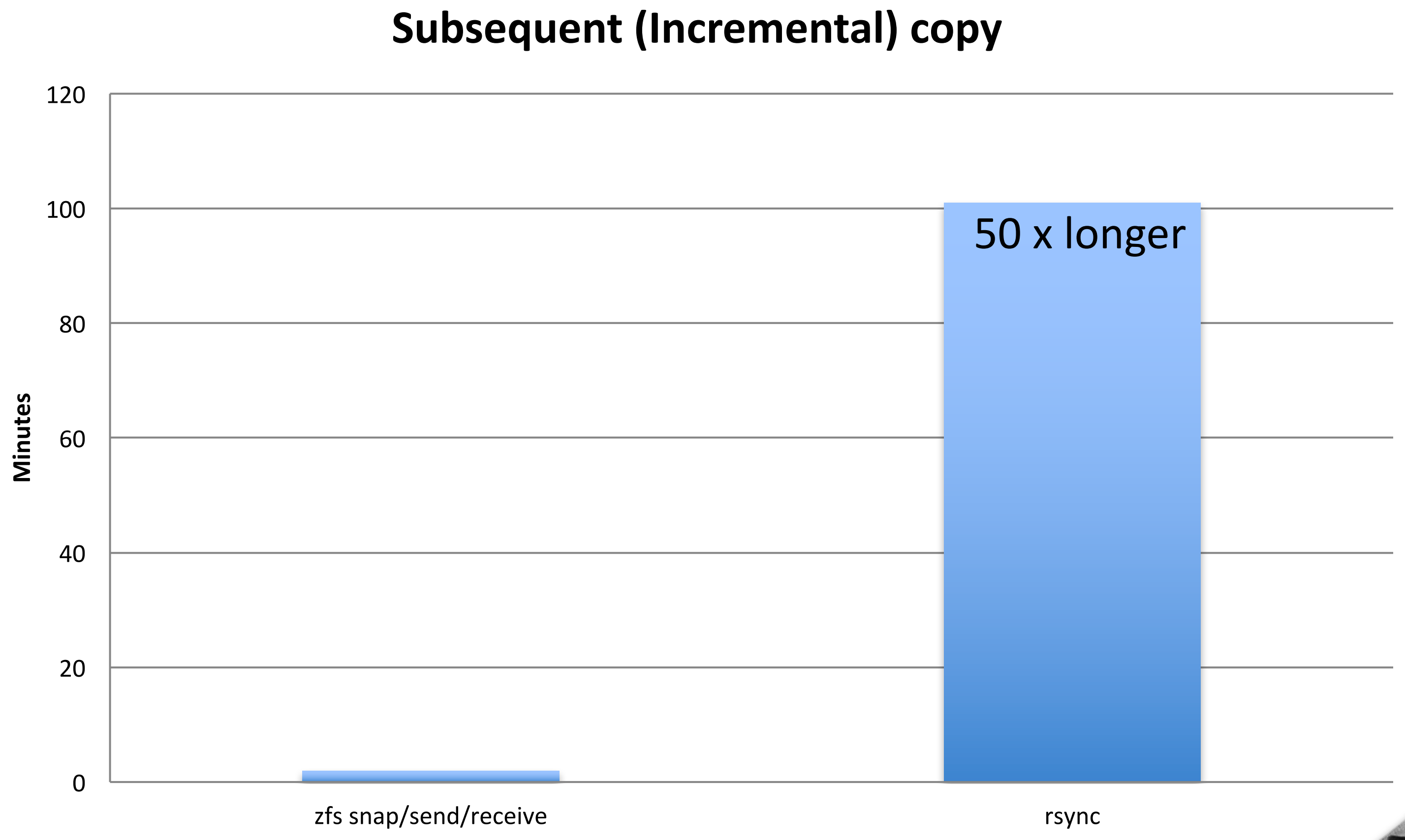
comparison on demo environment

Subsequent incremental rsync of the same data takes ~101 minutes

```
rsync -azh --no-whole-file . ${DEST}/
```

# Basic OST Migration – ZFS vs rsync

comparison on demo environment



# Lessons Learned

- Rsync is a great tool, but it has to “examine” every file which can be expensive with old hardware and/or millions or billions of files (lots of metadata calls)
- ZFS snap/send/receive operates underneath Lustre, there are zero metadata calls
- In a extremely high data rate of change (churn), incremental ZFS snap/send/receive may be inefficient. Snapshots can grow larger than initial OST, zpools can fill up.
- The right tool for the right job. ZFS snap/send/receive is another tool you can use, but might not be the right one.

# Futures

- Today, IU's production ZFS file system uses zpool concatenation of hardware raid devices. ZFS manages, but does not error correct.
- ZFS snap/send/receive will be used to migrate the above hardware protected OSTs, onto totally JBOD based zpools when hardware raid device is retired
- ZFS snap/send/receive will be used to migrate OSTs in between zpools, for performance balancing.
- ZFS snap/send/receive can be used for offline copy of an OST for further project research (Zester)

# References and Acknowledgements

- http://open-zfs.org/wiki/Main_Page
- https://pthree.org/2012/04/17/install-zfs-on-debian-gnulinux/

- Brian Behlendorf
- Chris Morrone
- Marc Stearman
- Andreas Dilger

# Questions?