BtrFS

Butter Filesystem. Hold the toast.

I've started experimenting with *BtrFS* which aims to provide an "advanced and modern filesystem" (heavily compared to ZFS) on Linux. With my new workstation I've started using BtrFS for my home directories (`/home`) and my build directories (`/mnt/slackbuilds`) to gain exposure to the filesystem and compare it to ZFS and EXT4 on LVM (all of my other data, including my root disk is on EXT4 on LVM).

I have used ZFS heavily in the past, and using BtrFS is significantly different as many of the fundamental concepts are different. BtrFS has no concept of "pools" or "volume groups" -- instead there are "volumes." BtrFS has no concept of "datasets" or "logical volumes" -- instead there are "subvolumes".

Here's a comparison between ZFS, BtrFS, and EXT4 on LVM:

	ZFS	BtrFS	EXT4 and LVM
Commands Involved	`zpool`, `zfs`	`mkfs.btrfs`, `btrfs`	`pvcreate`, `vgcreate`, `lvcreate`, `mkfs.ext4`
Pool of disks	"zpool"	"volume"	"volume group"
Mountable unit	"dataset"	"volume" and "subvolume"	"logical volume"
License	CDDL	GPL	GPL
Can be Boot filesystem	Yes	Yes (grub 2.00)	No
Can be Root filesystem	Yes	Yes	Yes
Can provide swapspace	Yes (zvols)	No	Yes (lvm)
OSes with Implementations	Solaris, OpenSolaris, Nexenta, FreeBSD, Mac OS X, Linux	Linux	Linux
Stability	Stable	Stable	Stable
CLI-System Integration [1]	Strong	Weak	Mild
Grow Online	Yes	Yes	Only when there are no snapshots
Shrink Pool	No	Online	Online
Shrink Filesystem	Online (reduce quota)	Online	Offline
Replace Disk (without	Yes (must be compatible	Yes	Yes (copies space allocated

	ZFS	BtrFS	EXT4 and LVM
parity)	size disk)		on disk)
Filesystem Level Storage Pooling	Yes	Yes	No
Re-balance	No	Yes	Can be done manually (`pvmove`)
Checksumming	Yes	Yes	No
Autocorrect Checksum Errors	Yes	???	No
Compression	Yes	Yes	No
De-duplication	Yes (only synchronous)	Yes (only asynchronous)	No
Ditto Blocks	Yes	???	No
Tiered Caching	Yes	No	No
Writable Snapshots	Yes (clone)	Yes	Yes
Copy-on-Write	Fast, space-efficient	Fast, space-efficient	Slow, requires pre-allocating an LV
Redundancy	Mirroring and Parity (x1, x2, x3)	Mirroring	Mirroring, though the PVs can be RAID devices
Maximum Volume Size	16 Exabytes	16 Exabytes	1 Exabyte
Maximum File Size	16 Exabytes	16 Exabytes	16 Terabytes
Maximum Number of Snapshots	*Unlimited*	*Unlimited*	Effectively 32

[1] For lack of a better term -- how well the command line interface integrates with the system as a whole, this might be subjective.

For a more complete, but less focused comparison see [Wikipedia's Comparison of Filesystems]

The Rosetta Stone

1. *Task:* Create a pool of storage from disks `/dev/A`, `/dev/B`, and `/dev/C` (striped or linear concat)
    a. Using ZFS:
        i. `# zpool create TESTPOOL A B C`
    b. Using BtrFS:
        i. `# mkfs.btrfs -d raid0 /dev/A /dev/B /dev/C`
    c. Using EXT4 on LVM:
        i. `# pvcreate /dev/A /dev/B /dev/C`
        ii. `# vgcreate TESTPOOL /dev/A /dev/B /dev/C`
2. *Task:* Make storage from pool available to system
    a. Using ZFS:
        i. `# zfs set mountpoint=/data TESTPOOL`
    b. Using BtrFS:
        i. `# mkdir /data`
        ii. `# mount -t btrfs /dev/A /data`
        iii. Update `/etc/fstab`
    c. Using EXT4 on LVM:
        i. `# mkdir /data`
        ii. `# lvcreate -L` *`SizeOfVolume`* `-n DATA TESTPOOL`
        iii. `# mkfs -t ext4 /dev/TESTPOOL/DATA`
        iv. `# mount /dev/TESTPOOL/DATA /data`
        v. Update `/etc/fstab`
3. *Task:* Add an additional disk to the pool
    a. Using ZFS:
        i. `# zfs add TESTPOOL D`
    b. Using BtrFS:
        i. `# btrfs device add /dev/D /data`
        ii. `# btrfs filesystem balance /data`
    c. Using EXT4 on LVM:
        i. `# pvcreate /dev/D`
        ii. `# vgextend TESTPOOL /dev/D`
4. *Task:* Add additional space to a filesystem
    a. Using ZFS:
        i. No action needed
    b. Using BtrFS:
        i. No action needed
    c. Using EXT4 on LVM:
        i. `# lvextend -L` *`SizeToIncreaseTo`* `/dev/TESTPOOL/DATA`
        ii. `# resize2fs /dev/TESTPOOL/DATA`

5. *Task:* Remove a disk from the pool
    a. Using ZFS: *N/A*
    b. Using BtrFS:
        i. `# btrfs device delete /dev/A /data`
        ii. `# btrfs filesystem balance /data`
    c. Using EXT4 on LVM:
        i. `# pvmove /dev/A`
        ii. `# vgreduce TESTPOOL /dev/A`
6. *Task:* Replace operational disk
    a. Using ZFS:
        i. `# zfs replace TESTPOOL A D`
    b. Using BtrFS:
        i. `# btrfs device add /dev/D /data`
        ii. `# btrfs device delete /dev/A /data`
        iii. `# btrfs filesystem balance /data`
    c. Using EXT4 on LVM:
        i. `# pvcreate /dev/D`
        ii. `# vgextend TESTPOOl /dev/D`
        iii. `# pvmove TESTPOOL /dev/A /dev/D`
        iv. `# vgreduce TESTPOOL /dev/A`
7. *Task:* Take a snapshot of a filesystem
    a. Using ZFS:
        i. `# zfs snapshot TESTPOOL@snapshot1`
    b. Using BtrFS:
        i. `# btrfs subvolume snapshot /data /data/snapshot1`
    c. Using EXT4 on LVM:
        i. `# lvcreate -s /dev/TESTPOOL/DATA -L SizeToAllocate -n snapshot1`
8. *Task:* Rollback a snapshot
    a. Using ZFS:
        i. `# zfs rollback TESTPOOL@snapshot1`
    b. Using BtrFS:
        i. *Not sure...*
    c. Using EXT4 on LVM:
        i. `# lvconvert --merge /dev/TESTPOOL/snapshot1`

9. *Task:* Delete a snapshot
    a. Using ZFS:
        i. `# zfs destroy TESTPOOL@snapshot1`
    b. Using BtrFS:
        i. `# btrfs subvolume delete /data/snapshot1`
    c. Using EXT4 on LVM:
        i. `# lvremove /dev/TESTPOOL/snapshot1`