ZFS on Linux

-

An Introduction

-

Cédric Dufour / cedric.dufour (AT) idiap.ch

ZFS on Linux
About the Speaker

- Cédric Dufour
- GNU/Linux System and Software Engineer
- ... at Idiap Research Institute (Martigny, VS)
- ... in charge of :
  - Network
  - Linux (Debian) hosts :
    - hypervisors (KVM)
    - servers
    - computation nodes
    - workstations

ZFS on Linux
Table of Content

ZFS on Linux
Filesystems on Linux

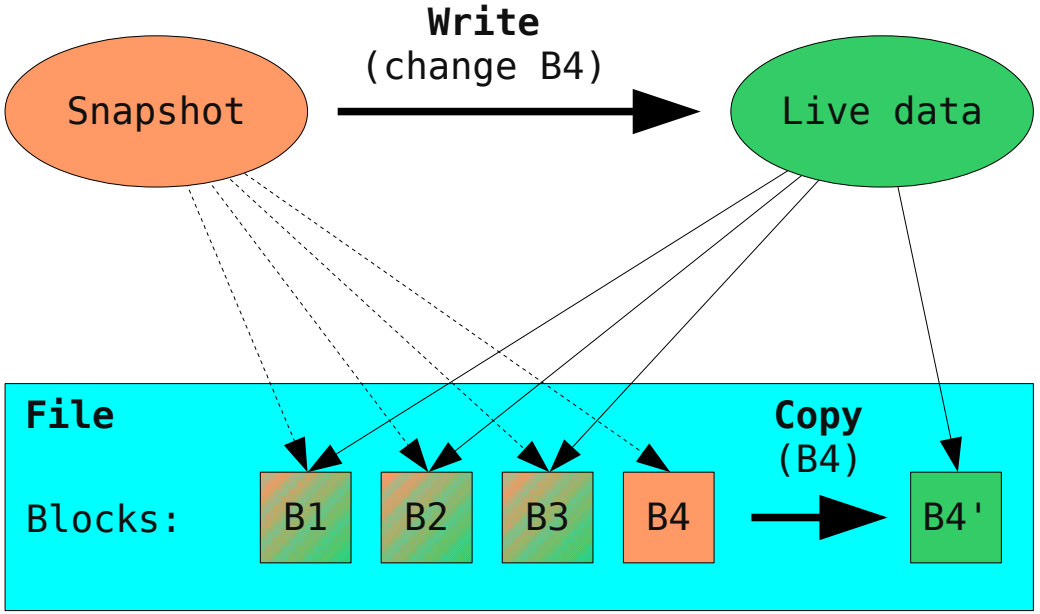
- The « incumbents »
  - ext2, ext3, ext4, ...
- The « alternatives »
  - xfs, reiserfs, ...
- The « rest of the world »
  - fat, ntfs, hfs, ...
- The « **newcomers** »
  - btrfs, **zfs**, ...

ZFS on Linux
ZFS, an old-timer

- Developed by Sun Microsystems since 2001 (Solaris)
- Open-sourced in 2005 (OpenSolaris)
  - Linux in 2005
  - Mac OS X in 2006
  - FreeBSD in 2007
- Closed-source by Oracle in 2010
  - *zpool* version 29 ongoing
- **OpenZFS since 2013** (illumos, FreeBSD, Linux, Mac OS X)
  - *zpool* version 28 + *feature flags*

ZFS on Linux
Copy-on-Write (COW)

- Copy-on-Write (COW)
  - pro: « naturally » goes along **snapshots**
  - con: induces fragmentation



ZFS on Linux
Checksum

- Checksum
  - **data integrity**
  - SHOULD (MUST!) use in all cases

```
$ zfs set checksum={on|fletcher4|...}
```

ZFS on Linux
« RAID »

- RAID, not really... but
- « Adaptive Block-Level » **data redundancy**
  - RAID0: striping (default)
  - RAID1: mirroring
    - RAID1+0: most practical (easy extension)
  - *RAIDZ-1* ~ RAID5: 1 parity block
  - *RAIDZ-2* ~ RAID6: 2 parity blocks
  - *RAIDZ-3*: 3 parity blocks
    - *RAIDZ-n*: most cost-saving
- self-healing (cf. checksum)

RAIDZ-1

Disk 1	Disk 2	Disk 3	Disk 4
A1	A2	A3	*Ap*
A4	A5	A6	*Ap'*
B1	*Bp*	C1	C2
C3	Cp	C4	*Cp'*

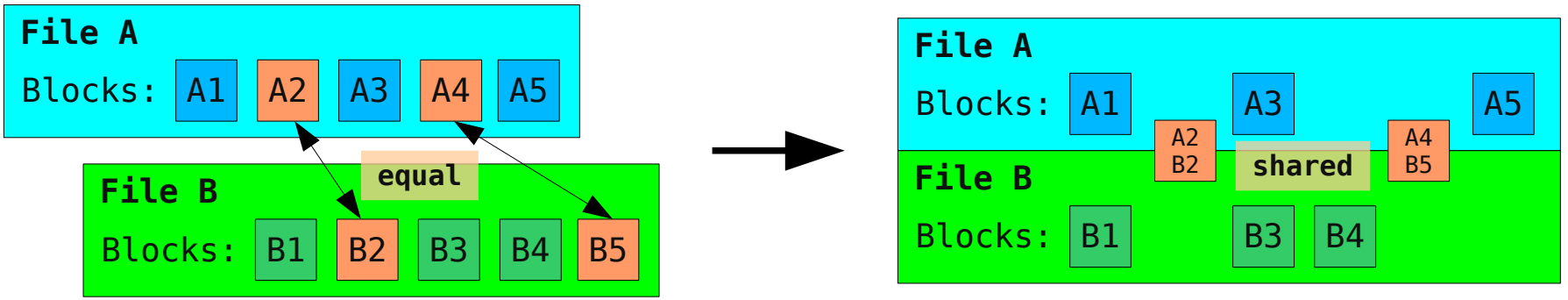
ZFS on Linux
Compression

- Compression
  - **capacity optimization**
  - « saved » I/Os do leverage the « wasted » CPU
  - SHOULD use in most cases

```
$ zfs set compression={on|lz4|...}
```

ZFS on Linux
Deduplication

- Deduplication
  - **capacity optimization**
  - uses a LOT of RAM (deduplication table)
  - unpractical for large storage capacity




```
$ zfs set dedup=on
```

ZFS on Linux
ARC / L2ARC

- « Adaptive Replacement Cache » (*ARC*)
  - RAM-based (by default up to 50% of available RAM)
- « Layer-2 Adaptive Replacement Cache » (*L2ARC*)
  - ideal along SSDs (or SAS 15k)
  - large "hybrid" cache
- Cache both recent reads and frequent reads
  - **read performances**

```
$ zfs set primarycache={all|metadata|none}   # ARC
$ zfs set secondarycache={all|metadata|none} # L2ARC
```

ZFS on Linux
ZIL / SLOG

- « ZFS Intent Log » (*ZIL*)
  - write journal
  - **data integrity**
- « Separate Intent Log » (*SLOG*)
  - ideal along SSDs (or SAS 15k)
  - only for synchronous writes
  - **write performances**

```
$ zpool add … log … # SLOG
```

ZFS on Linux
« pros »

- COW (Copy-on-Write)
  - **snapshots**
- checksum (block-level)
  - **data integrity**
- *raidz* (block-level)
  - **data redundancy**
- compression (& deduplication)
  - save physical space
- *L2ARC* (read cache; NVRAM/SSD/SAS)
  - read performances
- *SLOG* (write journal; NVRAM/SSD/SAS)
  - write performances

ZFS on Linux
« cons »

- Copy-on-Write (COW)
  - fragmentation
- depuplication
  - unpractical (very high RAM usage)
- encryption
  - not natively (use LUKS on top of *zdev*)
- O_DIRECT
  - not supported (use *sync=always*)
- Common Development and Distribution License (CDDL)
  - GPL-incompatible

ZFS on Linux
Installation

- **http://zfsonlinux.org**
  - Debian, Arch, Gentoo, Fedora, ...
- Two components (kernel modules)
  - Solaris Porting Layer (SPL)
  - ZFS itself
- *ARC* vs RAM (by default up to 50% of available RAM)
  - /etc/modprobe.d/zfs.conf (*zfs_arc_min* / *zfs_arc_max*)
- boot (/boot/...) partition
  - not possible (until grub gets ZFS write ability)
- root (/) partition
  - possible but not straight-forward (see the FAQ)

ZFS on Linux
ZFS vs LVM

- Physical devices
  - LVM/ZFS: SATA, SAS, SSD, NVRAM (/dev/...)
- Physical aggregation
  - LVM: Volume Group (vg...)
  - ZFS: *pool* (*zpool*)
- Logical organization
  - LVM: Logical Volume (lv...)
  - ZFS: *filesystem* (*zfs*)

ZFS on Linux
zpool command

- *zpool ...*
  - Create the *pool*(s)
    - aggregate (stripe, mirror, raidz) physical devices in *VDEVs*
    - each *pool* can have multiple *VDEVs*
    - a *pool* can be dynamically extended (add *VDEVs*)
  - Create the *L2ARC* (2nd-level read cache)
  - Create the *SLOG* (2nd-level write journal)

```
$ zpool create ${pool} raidz1 sd${X} sd${Y} sd${Z} # pool
$ zpool add ${pool} cache sd${U} sd${V}            # L2ARC
$ zpool add ${pool} log mirror sd${R} sd${S}       # SLOG
$ man zpool
```

ZFS on Linux
Configuration & status

- *zpool set … / zpool get …*
  - Self-contained configuration and status information, for each *pool*:
    - *size / capacity / free / allocated / fragmentation*
    - *autoreplace / autoexpand*
    - *feature@…*
    - etc.

```
$ zpool get all ${pool}  # show entire configuration
$ zpool get … ${pool}    # get configuration parameter
$ zpool set … ${pool}    # set configuration parameter
```

ZFS on Linux
zfs command

- *zfs …*
  - Create the filesystem(s)
  - Configure the filesystem(s)
  - Create/delete snapshots
  - Send/receive datasets/snapshots (« rsync »)

```
$ zfs create ${filesystem}                              # filesystem
$ zfs set … ${filesystem}                               # config
$ zfs snapshot|destroy ${filesystem}@${snapname} # snapshot
$ zfs send|receive … ${filesystem}@${snapname}   # “rsync”
$ man zfs
```

ZFS on Linux
Configuration & status

- *zfs set … / zfs get …*
  - Hierarchical, self-contained configuration and status information, for each *filesystem*:
    - *used* / *available* [capacity] / *compressratio*
    - *checksum* / *compression* / *dedup*
    - *readonly* / *atime* / *relatime* / *xattr* / *sync* / *exec* / *setuid*
    - *primarycache* / *secondarycache*
    - etc.

```
$ zfs get all ${filesystem}  # show entire configuration
$ zfs get … ${filesystem}    # get configuration parameter
$ zfs set … ${filesystem}    # set configuration parameter
```

ZFS on Linux
Maintenance

- *zpool …*
  - Check the pool status
  - Check the pool data integrity (*scrub*)
  - Replace a physical device
  - View the entire ZFS (commands) history (!)

```
$ zpool status ${pool}     # status
$ zpool scrub ${pool}      # scrub (check data integrity)
$ zpool replace ${pool} … # replace device
$ zpool history            # history
```

ZFS on Linux
Conclusion

- Great features
  - **enterprise-grade storage**
- Easy to install
  - packages available for most distributions
- Easy to maintain
  - powerful yet simple-enough command-line utilities
  - exhaustive documentation and supportive community
- Stability, reliability and performances
  - 1-year personal experience @ home : no problemo
  - https://github.com/zfsonlinux/zfs/issues
- Bottom-line...
  - **... give it a try !!!**

ZFS on Linux
References

- The official multi-platform open source ZFS initiative
  - **http://open-zfs.org**
- Linux-specific ZFS ↔ ZFS on Linux (ZoL)
  - **http://zfsonlinux.org**
- An excellent reference on ZFS inside-outs
  - Aaron Toponce's ZFS serie **https://pthree.org/category/zfs/**