Explain these results

```
naxos mge ~ $ find /mnt/ -type f -size -1k | wc -l
 45101
```

```
naxos mge ~ $ find /mnt/ -type f -size -1024c | wc -l
 107058
```

Comparing Filesystems

Matthias G. Eckermann
Senior Product Manager
SUSE Linux Enterprise
mge@suse.com

Vojtech Pavlik
Director SUSE Labs

vojtech@suse.com

SUSECon 2015
2015-10-29 11:06 UTC

Comparing Filesystems

With a little bit more focus on btrfs than originally planned

Matthias G. Eckermann
Senior Product Manager
SUSE Linux Enterprise
mge@suse.com

Vojtech Pavlik
Director SUSE Labs

vojtech@suse.com

SUSECon 2015
2015-10-29 11:06 UTC

Let's start ...

… where the idea of this presentation started.

Let's start ...

… where the idea of this presentation started.

snapper rollback …

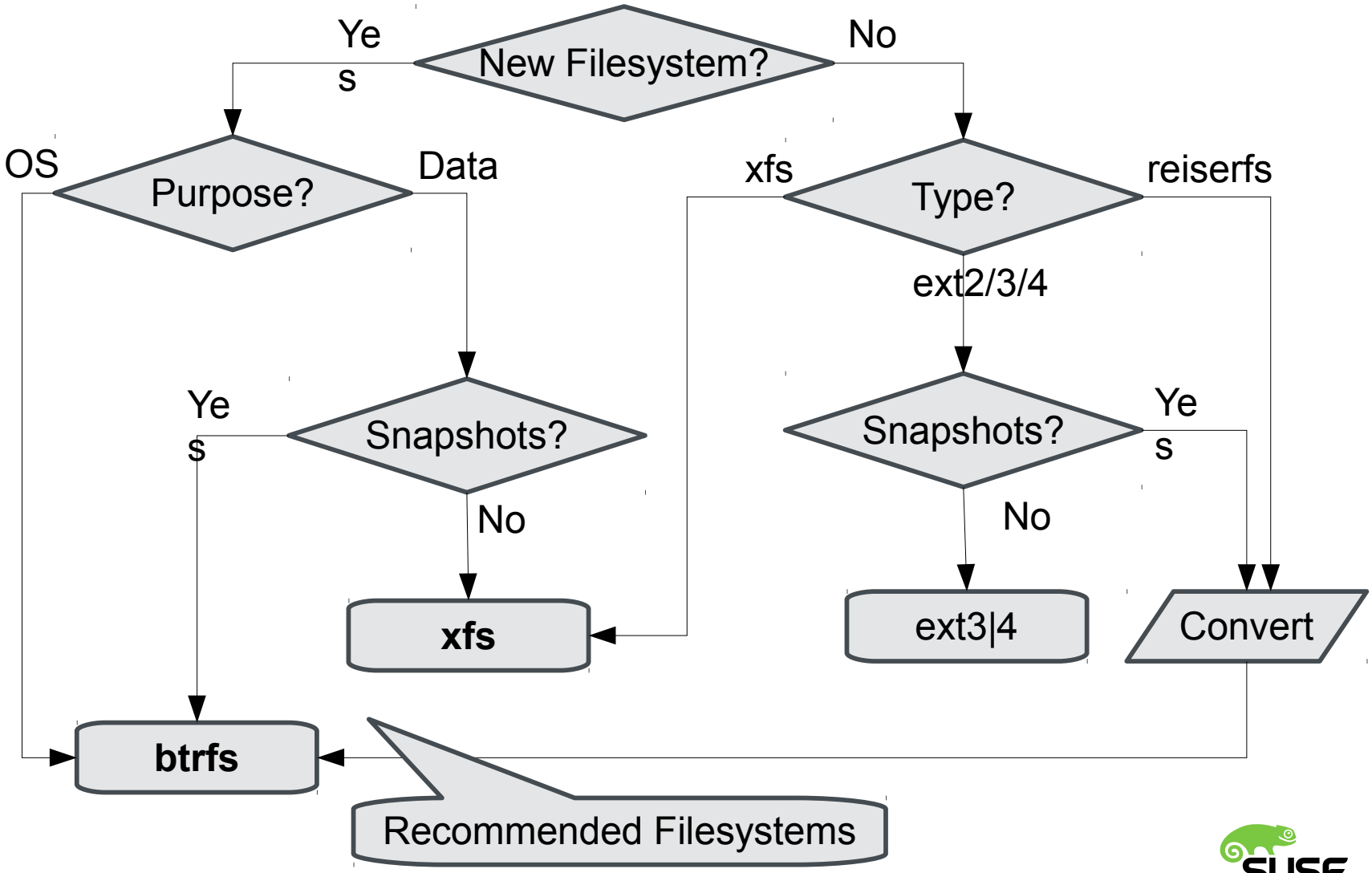
Let's start ...

… where the idea of this presentation started.

SUSECon 2014

SUSE® Linux Enterprise 12

Filesystem recommendations



Why …

… these recommendations?

Comparing Filesystems

Agenda

Space Efficiency

50 000 000 files

Performance

Features

Space Efficiency

A SUSE internal discussion some time ago ...

T: I hate btrfs.

M: Why?

T: It's a space hog beyond comparison.

M: Indeed, with snapshots it is. But that's a feature. Without snapshots it is as efficient as reiserfs.

T: Huh? All around me are wondering: what does M. smoke to believe that btrfs is as efficient as reiserfs?

M. is confused and starts a comparison, ...

Dataset #1
/home/mge/projects

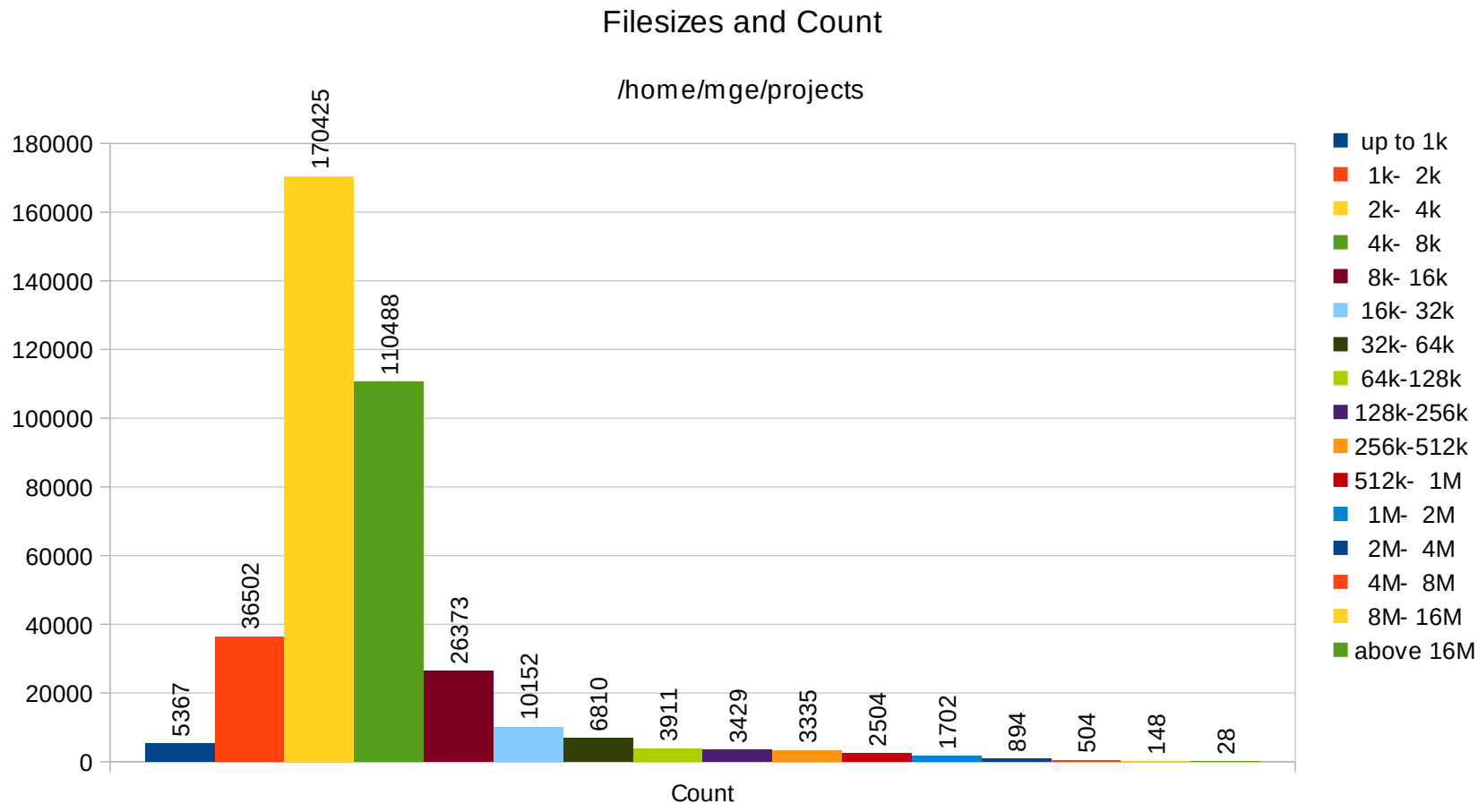
Filesystem Space Efficiency

Dataset#1 – Test parameters

- Partition size: 24626 MiB, SSD
- Original data: E-Mails and Documents
- Original data is in an encrypted partition and copied using “rsync -aPHSvu /source /target”

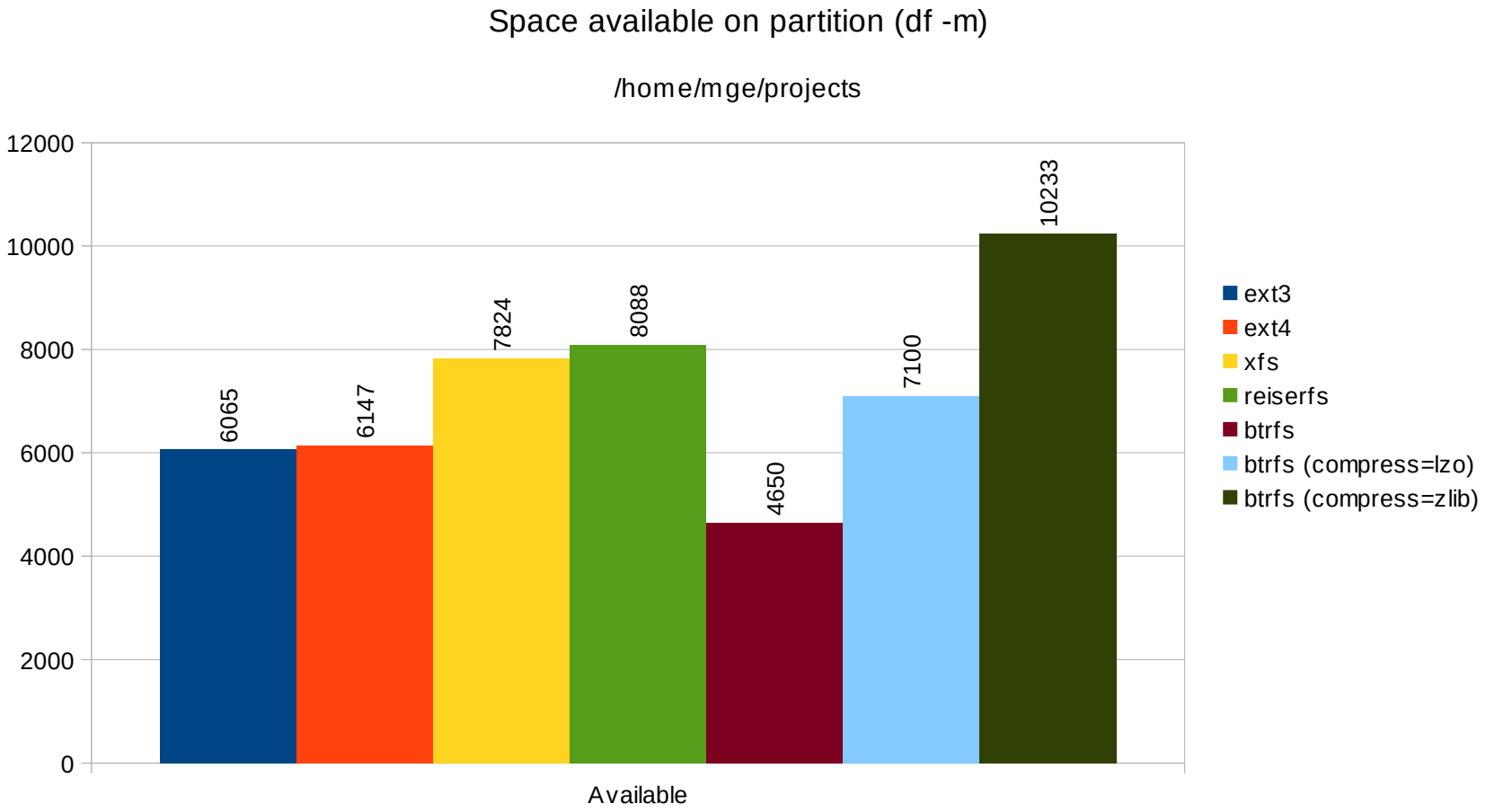
Filesystem Space Efficiency

Dataset#1 – Data typology



Filesystem Space Efficiency

Dataset#1 – Result



btrfs *is* a space hog.

xfs and reiserfs are leading head to head.

btrfs with zlib compression is space efficient as well. Mind the cpu overhead though.

Result

Dataset #2
/space

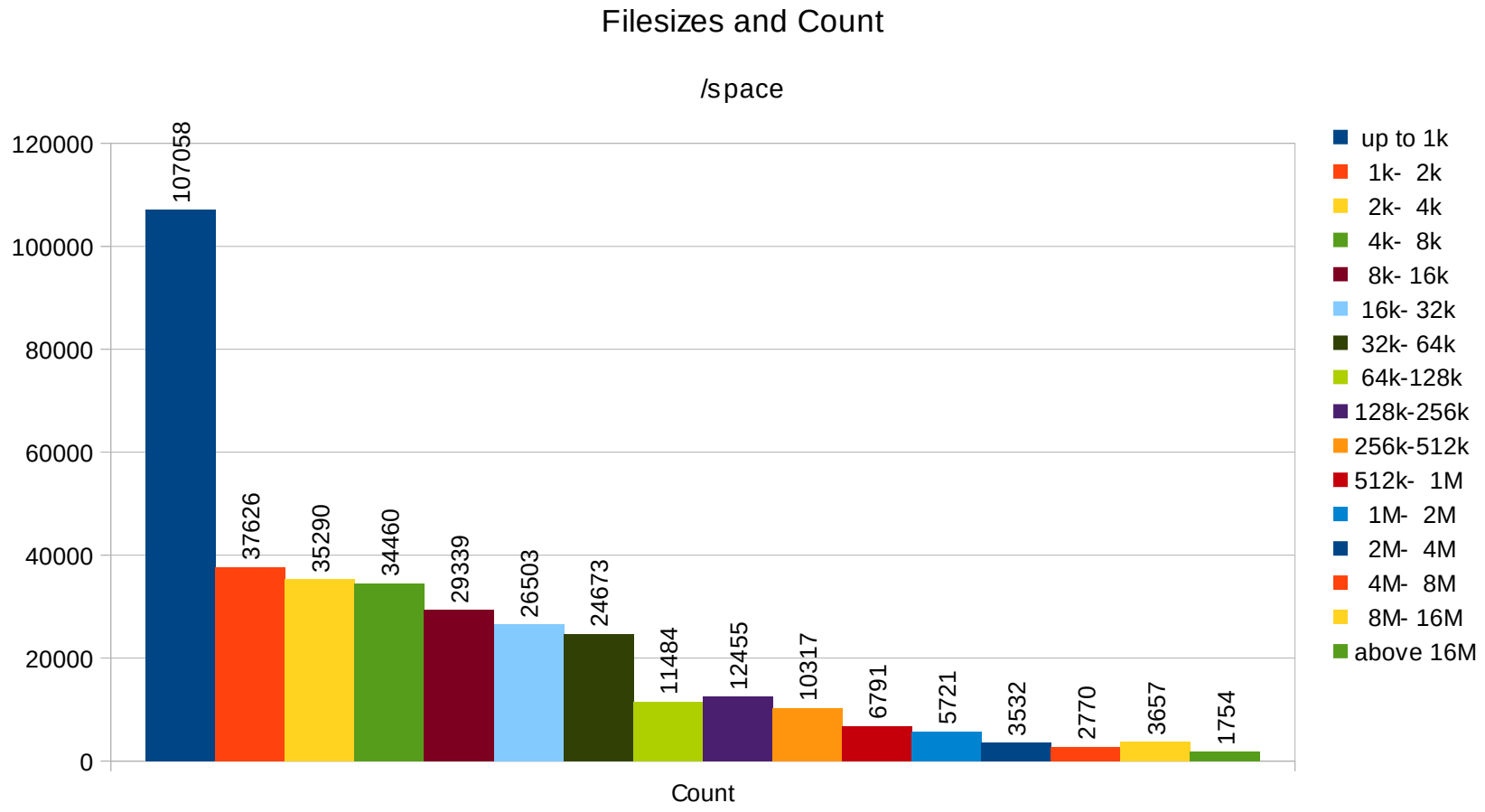
Filesystem Space Efficiency

Dataset#2 – Test parameters

- Partition size: 303884 MiB, HDD
- Original data
  - ISOs from various OSs (~100 GiB)
  - SLE package mirror (RPMs) (~50 GiB)
  - VMs (~25 GiB)
  - Pictures and Sounds (~20 GiB)
  - Third party software including TeXLive (~16 GiB)
  - /usr/src (including expanded Linux Kernel trees) and self-compiled RPMs (~16 GiB)
- Original data is copied using "rsync -aPHSvu /source /target"
- Tested on xfs and btrfs only

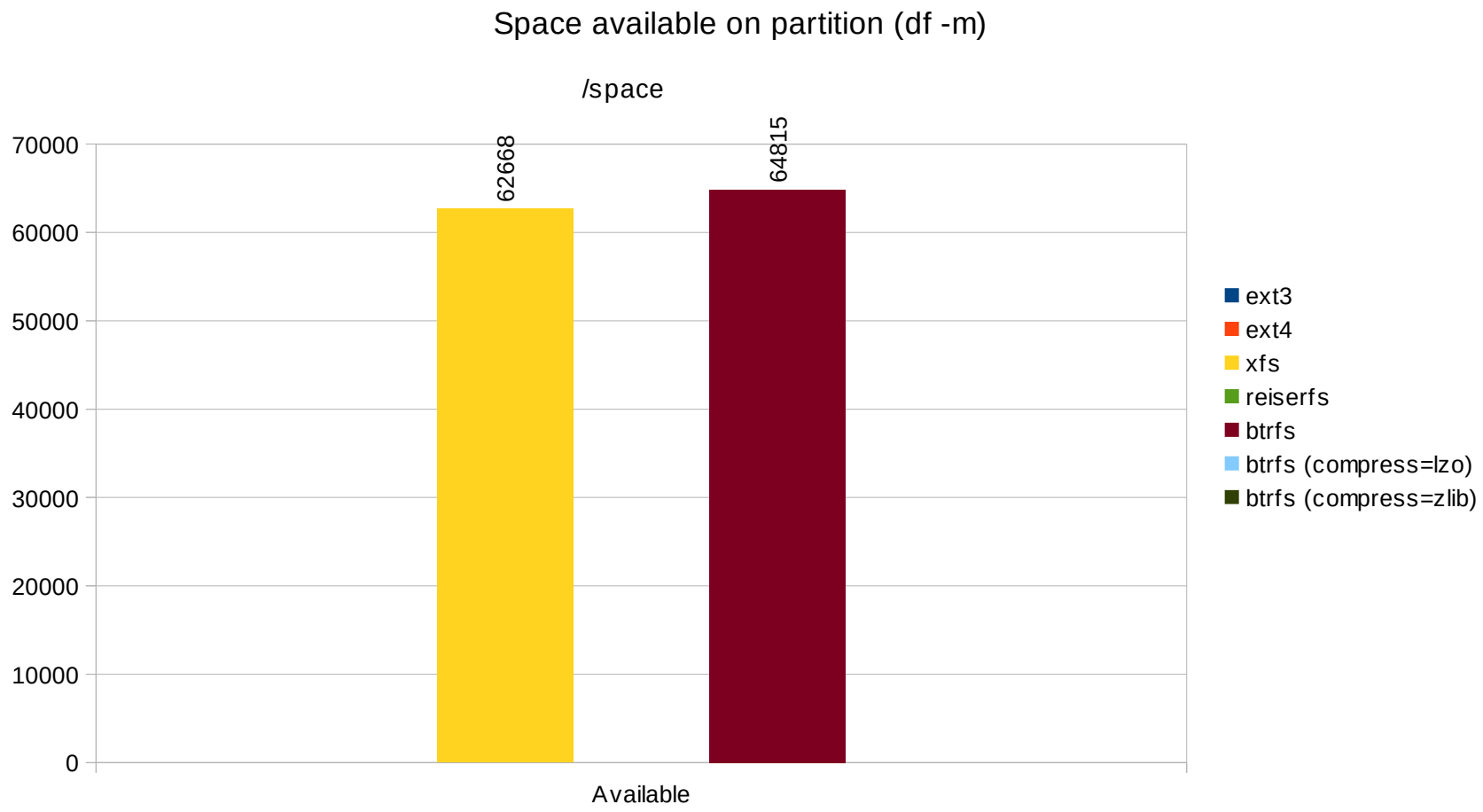
Filesystem Space Efficiency

Dataset#2 – Data typology



Filesystem Space Efficiency

Dataset#2 – Result



btrfs is *not* a space hog,
but more efficient then xfs.

Result

Final Result

Don't trust any statistics
you didn't make up yourself!

50 000 000 Files

Customer Question

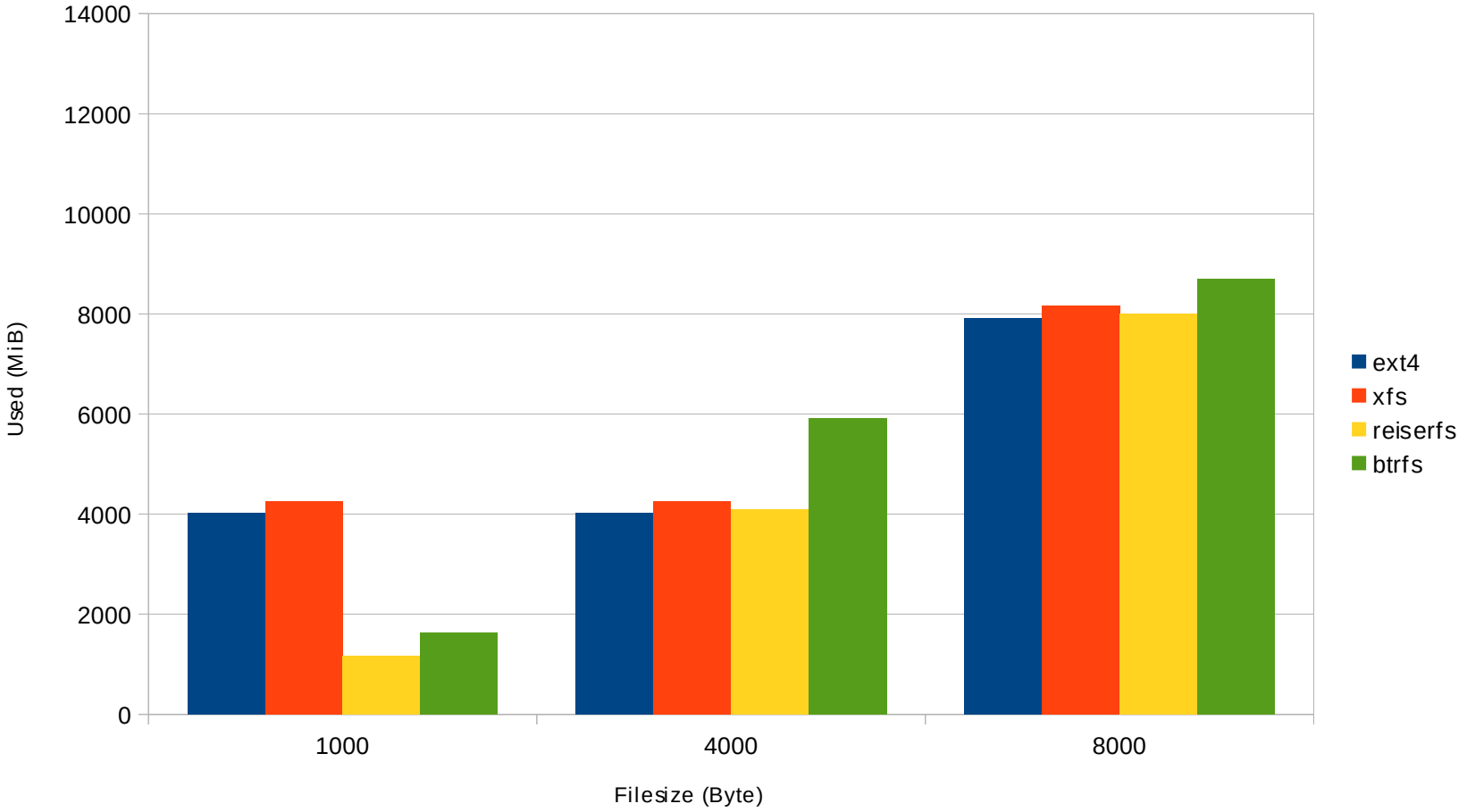
Which is the best filesystem to store 50 000 000 very small files on it?

Very small = ~1 KiB

Customer vertical: telecommunications

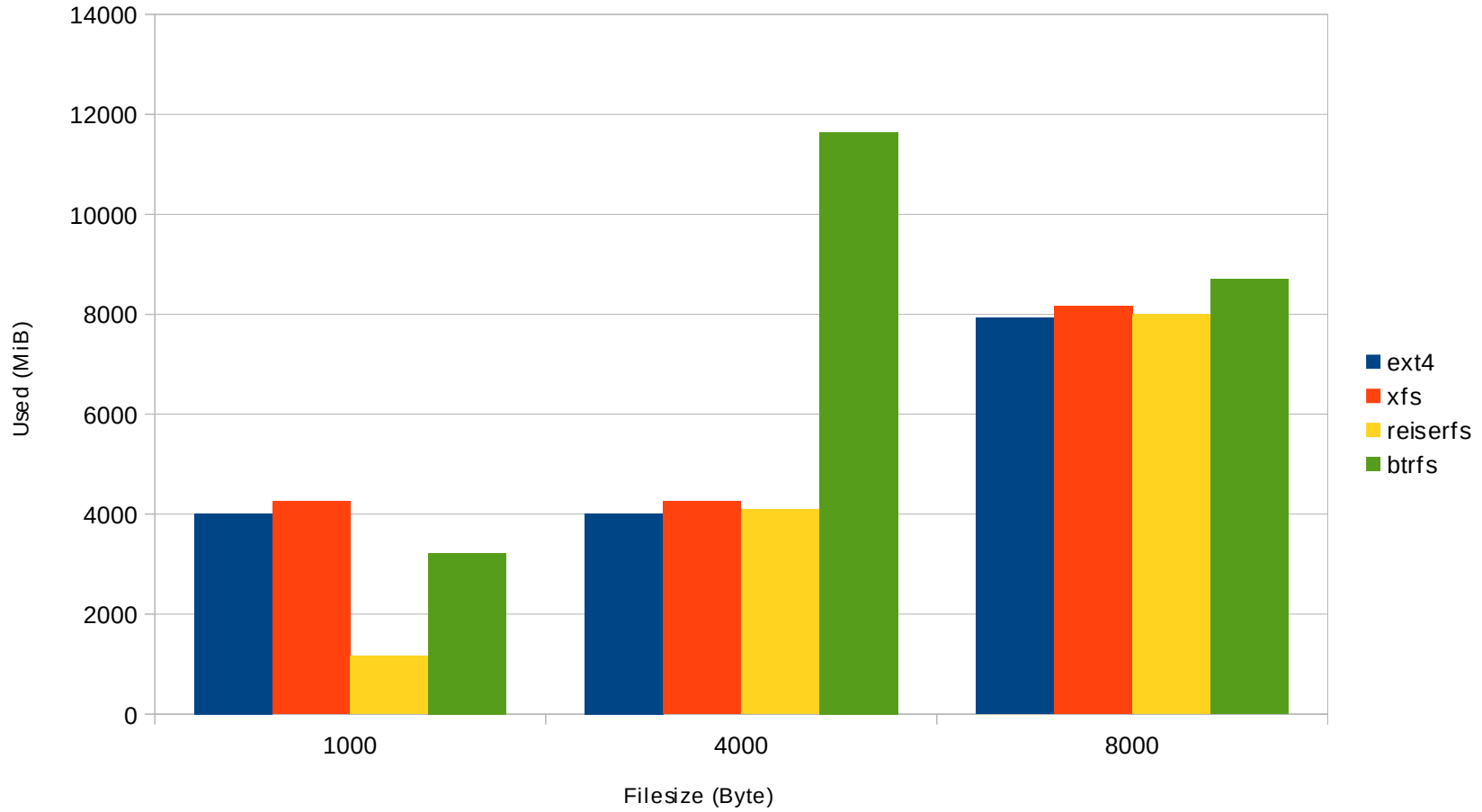
Comparing Filesystems

Disk Usage (1 million files, SSD)



Comparing Filesystems

Disk Usage (1 million files, HDD)



Comparing Filesystems

Disk Usage

- xfs / ext4
  - Expected Results
- Reiserfs
  - Expected Results (tail packing)
- btrfs
  - Expected results for 1000 and 8000 Bytes SSD
  - Suprising results on HDD, specifically at 4000 Bytes

And now, you better call a kernel developer you trust, ….

Comparing Filesystems

Disk Usage

… and he starts talking about “Metadata”

Comparing Filesystems

btrfs filesystem usage (1000 Byte, SSD)

```
Overall:
    Device size:                  14.87GiB
    Device allocated:              3.27GiB
    Device unallocated:           11.60GiB
    Used:                          1.56GiB
    Free (estimated):             12.61GiB      (min: 12.61GiB)
    Data ratio:                    1.00
    Metadata ratio:                1.00
    Global reserve:               32.00MiB      (used: 0.00B)

Data,single: Size:1.01GiB, Used:576.00KiB          /dev/sdb1       1.01GiB
Metadata,single: Size:2.26GiB, Used:1.56GiB        /dev/sdb1       2.26GiB
System,single: Size:4.00MiB, Used:16.00KiB         /dev/sdb1       4.00MiB
Unallocated:                                       /dev/sdb1      11.60GiB
```

Comparing Filesystems

btrfs filesystem usage (4000 Byte, SSD)

```
Overall:
    Device size:                  14.87GiB
    Device allocated:              7.27GiB
    Device unallocated:            7.60GiB
    Used:                          5.64GiB
    Free (estimated):              8.61GiB      (min: 8.61GiB)
    Data ratio:                    1.00
    Metadata ratio:                1.00
    Global reserve:              128.00MiB      (used: 0.00B)

Data,single: Size:1.01GiB, Used:1.56MiB          /dev/sdb1     1.01GiB
Metadata,single: Size:6.26GiB, Used:5.64GiB      /dev/sdb1     6.26GiB
System,single: Size:4.00MiB, Used:16.00KiB       /dev/sdb1     4.00MiB
Unallocated:                                     /dev/sdb1     7.60GiB
```

Comparing Filesystems

btrfs filesystem usage (4000 Byte, HDD)

```
Overall:
    Device size:                  14.87GiB
    Device allocated:             13.04GiB
    Device unallocated:            1.84GiB
    Used:                         11.25GiB
    Free (estimated):              2.84GiB      (min: 1.92GiB)
    Data ratio:                    1.00
    Metadata ratio:                2.00
    Global reserve:              128.00MiB      (used: 0.00B)

Data,single: Size:1.01GiB, Used:2.69MiB          /dev/sdb1       1.01GiB
Metadata,single: Size:8.00MiB, Used:0.00B        /dev/sdb1       8.00MiB
Metadata,DUP: Size:6.00GiB, Used:5.62GiB         /dev/sdb1      12.00GiB
System,single: Size:4.00MiB, Used:0.00B          /dev/sdb1       4.00MiB
System,DUP: Size:8.00MiB, Used:16.00KiB          /dev/sdb1      16.00MiB
Unallocated:                                     /dev/sdb1       1.84GiB
```

Comparing Filesystems

btrfs filesystem usage (8000 Byte, SSD)

```
Overall:
    Device size:                  14.87GiB
    Device allocated:              9.27GiB
    Device unallocated:            5.60GiB
    Used:                          8.31GiB
    Free (estimated):              5.98GiB      (min: 5.98GiB)
    Data ratio:                    1.00
    Metadata ratio:                1.00
    Global reserve:              192.00MiB      (used: 0.00B)

Data,single: Size:8.01GiB, Used:7.63GiB          /dev/sdb1     8.01GiB
Metadata,single: Size:1.26GiB, Used:697.94MiB    /dev/sdb1     1.26GiB
System,single: Size:4.00MiB, Used:16.00KiB       /dev/sdb1     4.00MiB
Unallocated:                                     /dev/sdb1     5.60GiB
```

Btrfs Metadata

Integrity & Resilience

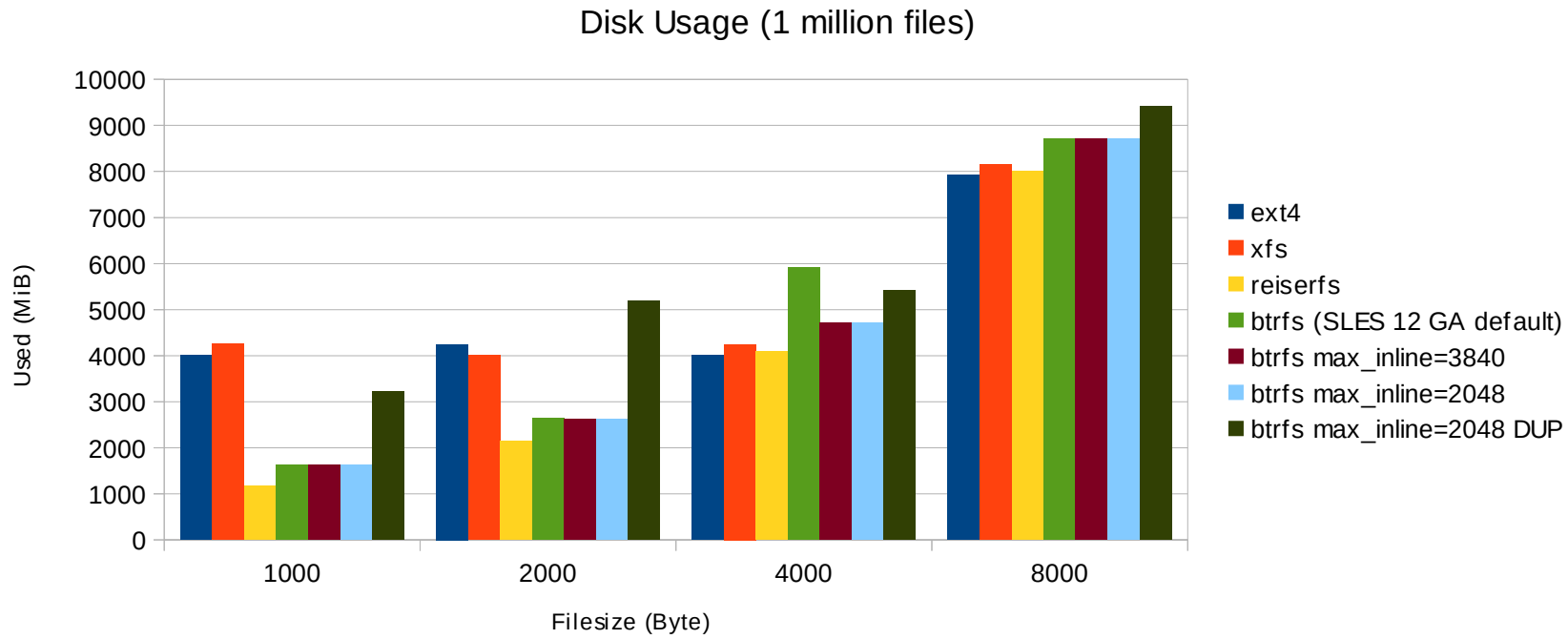
- Metadata duplication
- Back references

“Efficiency”

- Files < Blocksize are stored in metadata space

Comparing Filesystems

Btrfs Metadata



Btrfs Metadata – SLE 12 SP1

Change for SUSE Linux Enterprise 12 SP1

Files < 2048 Byte (half blocksize) are stored in metadata

→ Significant space savings

Performance

Performance – Goals of this discussion

NOT

- Compare performance of filesystems

BUT

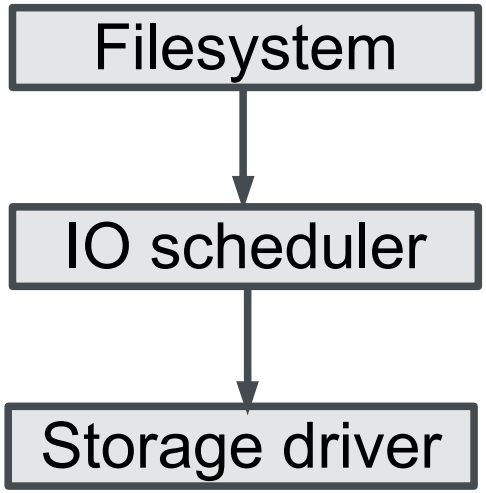
- Show, which parameters influence performance
- Find “characteristics” of filesystems (if any)

I/O Scheduler

Comparing Filesystems

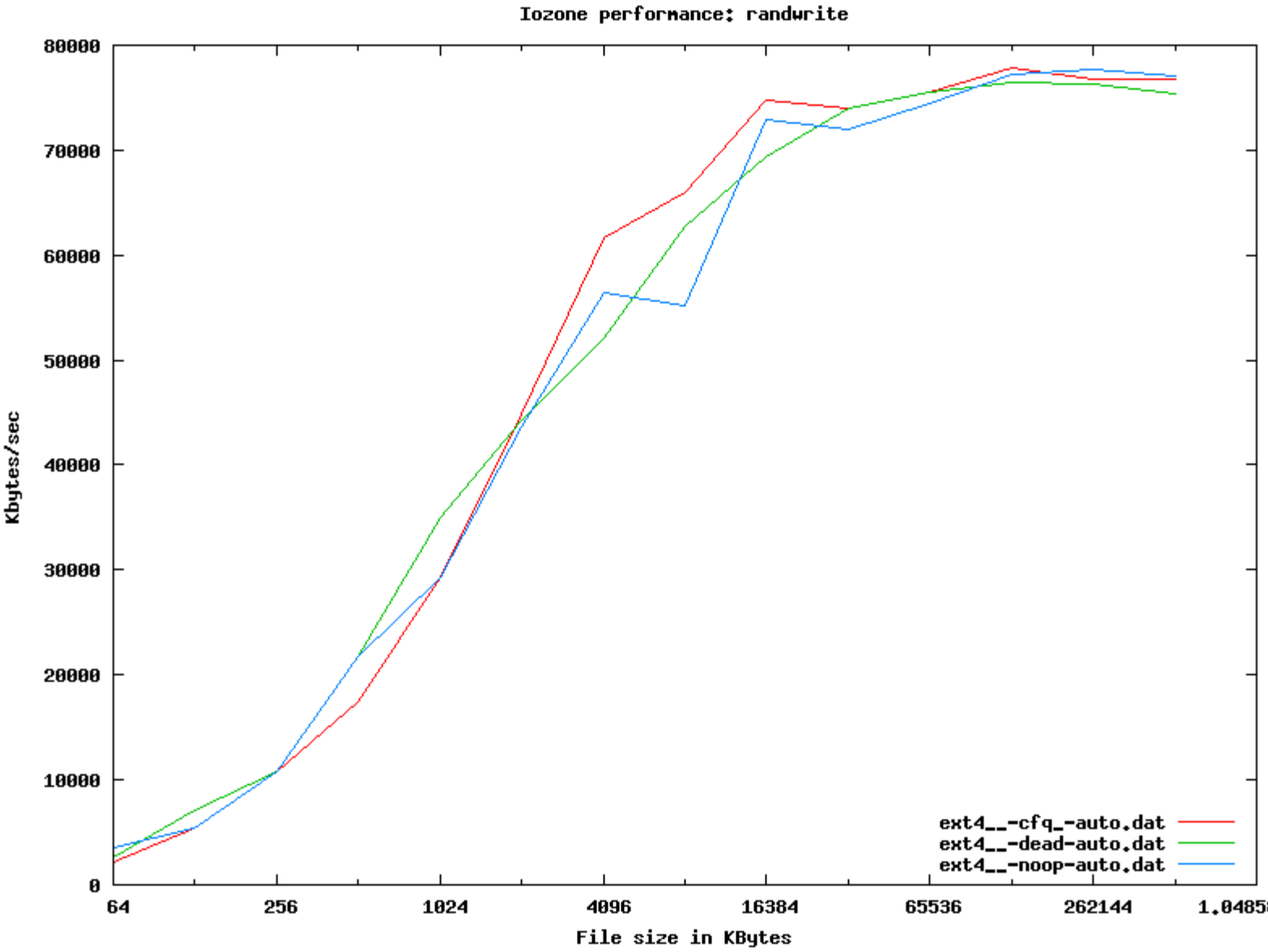
I/O - Scheduler

- Get requests from file system, memory management, etc. <*dev, start, len*>
- Pass requests further to device drivers
- Three IO schedulers – Noop, Deadline, CFQ



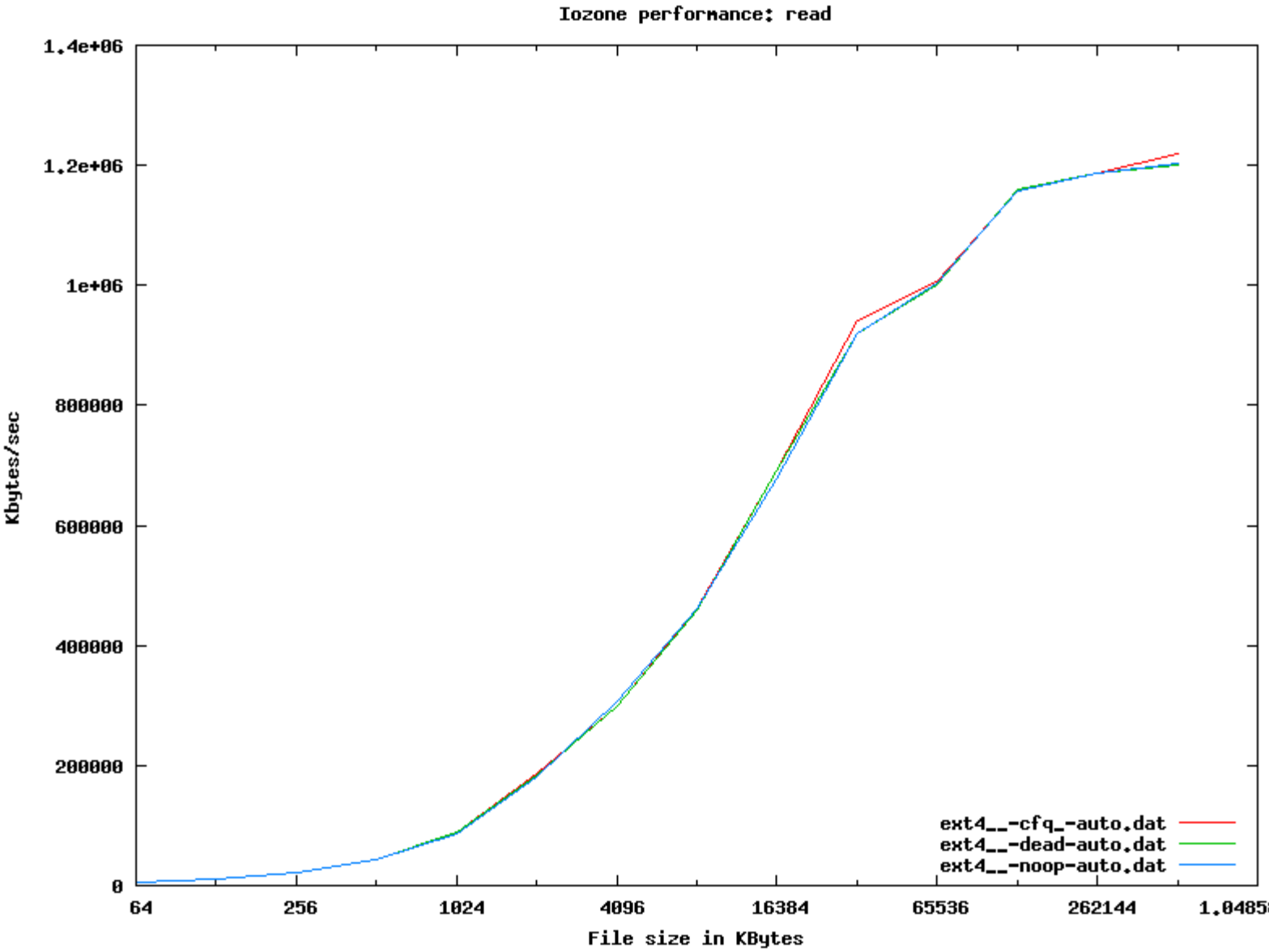
Comparing Filesystems

I/O-Schedulers (HDD², ext4 only)



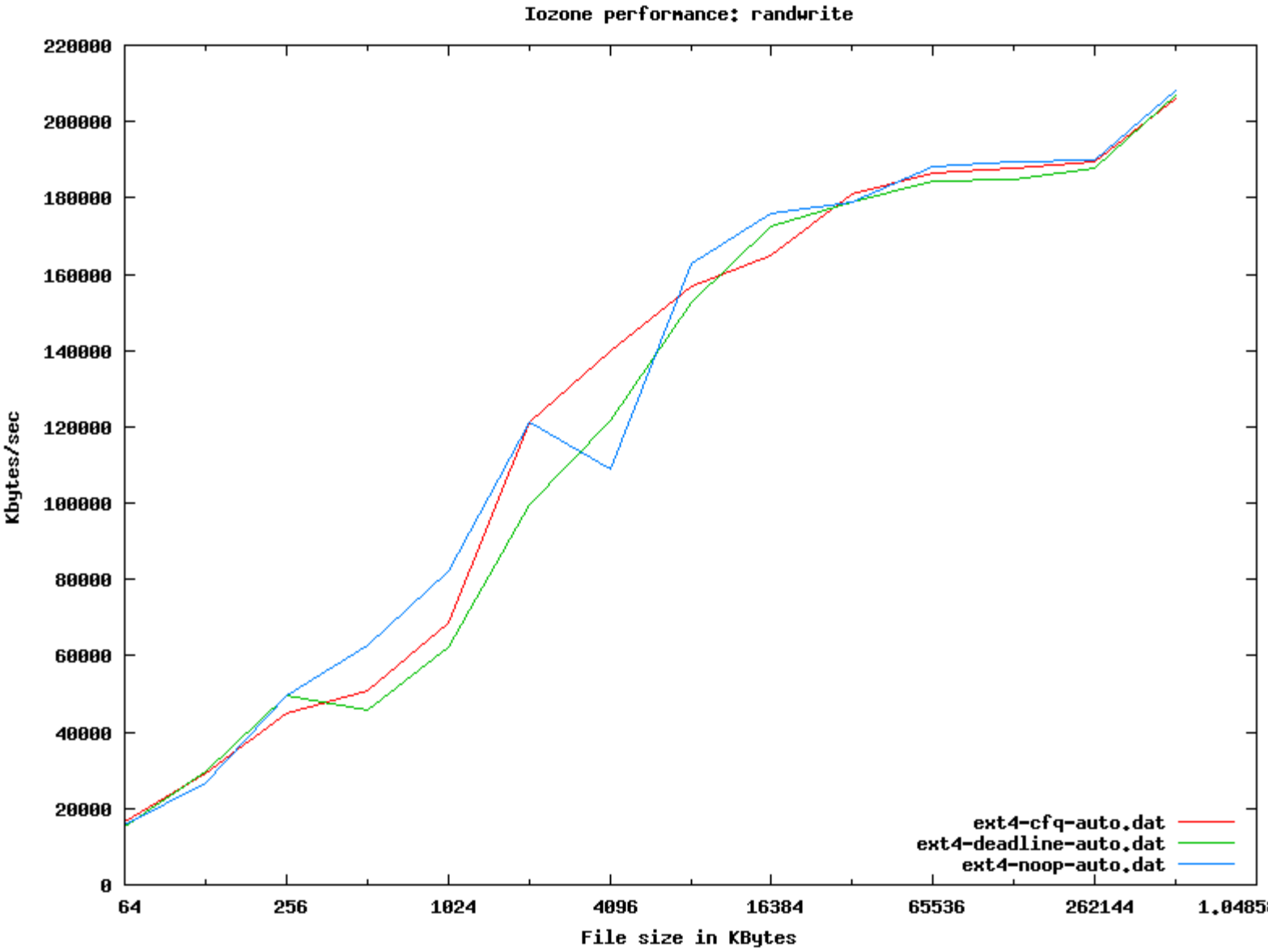
Comparing Filesystems

I/O-Schedulers (HDD², ext4 only)



Comparing Filesystems

I/O-Schedulers (SSD[1], ext4 only)



I/O-Schedulers – “cfq”

The cfq scheduler (completely fair queuing) should be used, if you want to

- Achieve a *fair* distribution of time slices among processes/ threads.
- Tune the scheduler according to your infrastructure, application and needs

Single threads can achieve high throuput in sequential writes → certain risk of starvation for other threads.

cfq is default on SUSE Linux Enterprise.

I/O-Schedulers – “noop”

The “noop” scheduler should be used, if the storage backend has its own scheduler. This is true for

- Multipathing
- Virtual machine guests
- High-end flash devices

I/O-Schedulers – “deadline”

The “deadline” scheduler should be used in cases where you need to avoid *starvation* of processes/threads:

- Databases
- VM Hosts, e.g. KVM or Xen

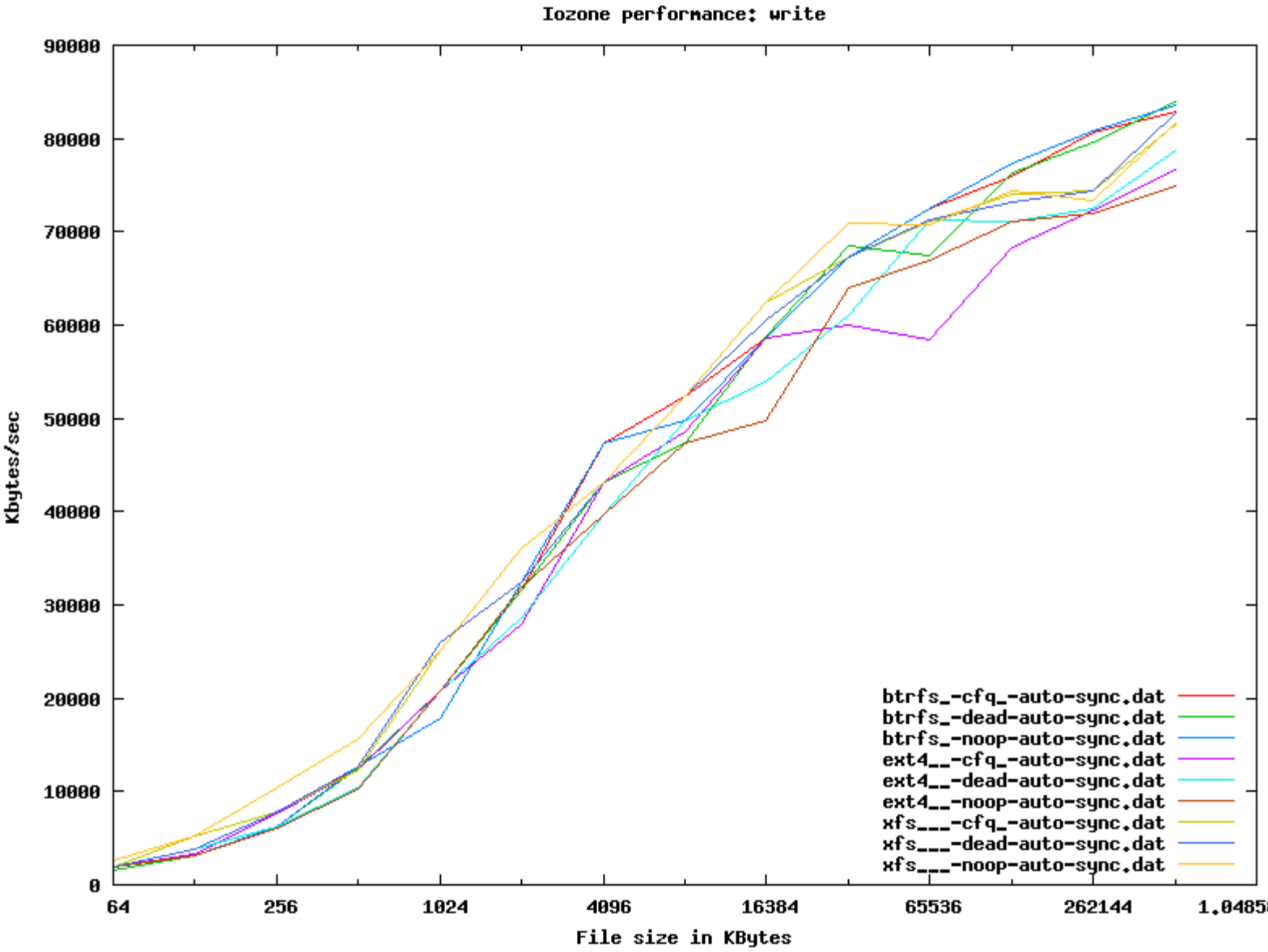
Deadline is also the most suitable for

- Consumer grade flash devices (SSD)

Storage Backend

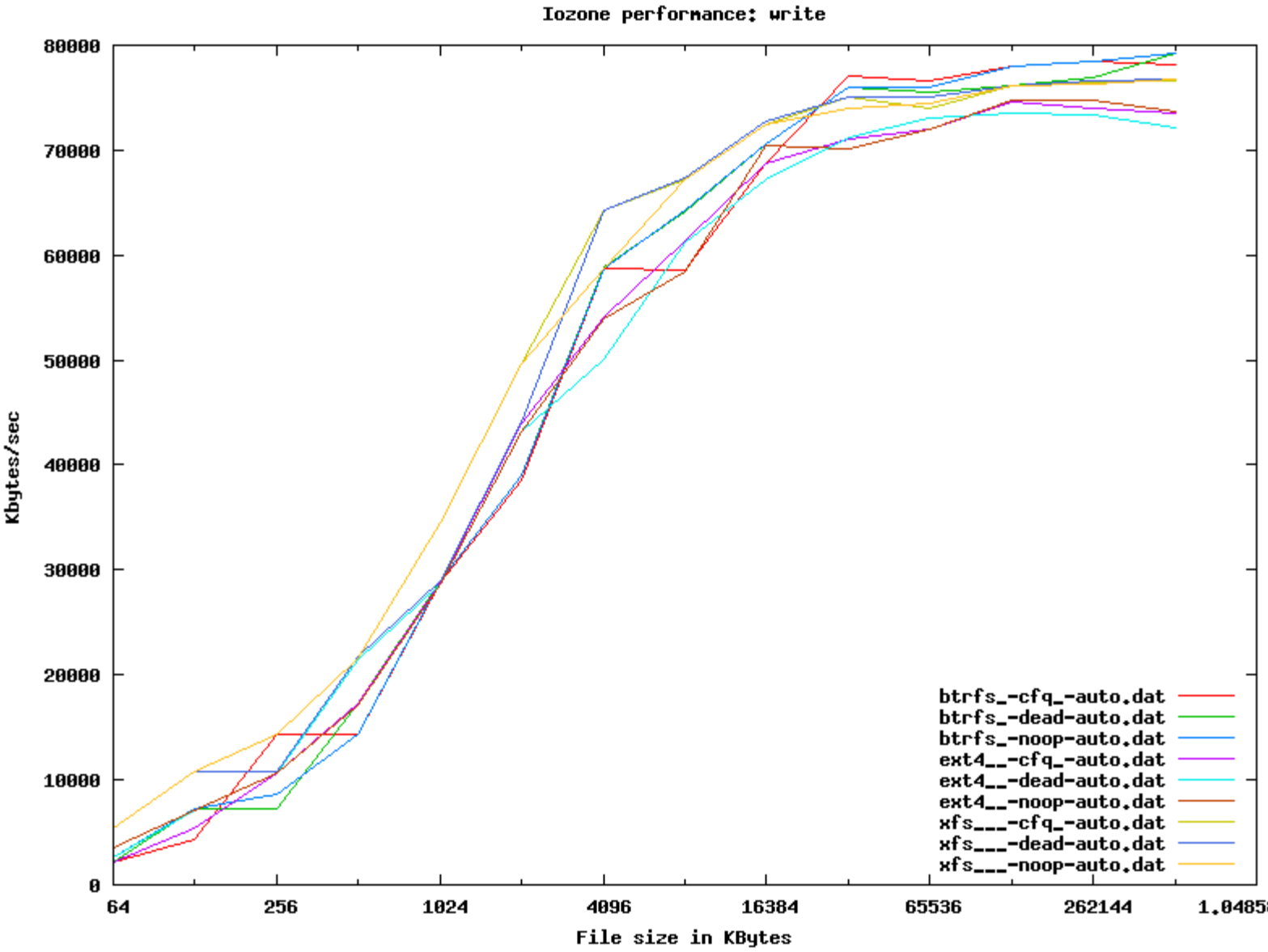
Comparing Filesystems

Storage Backend HDD[1] – write



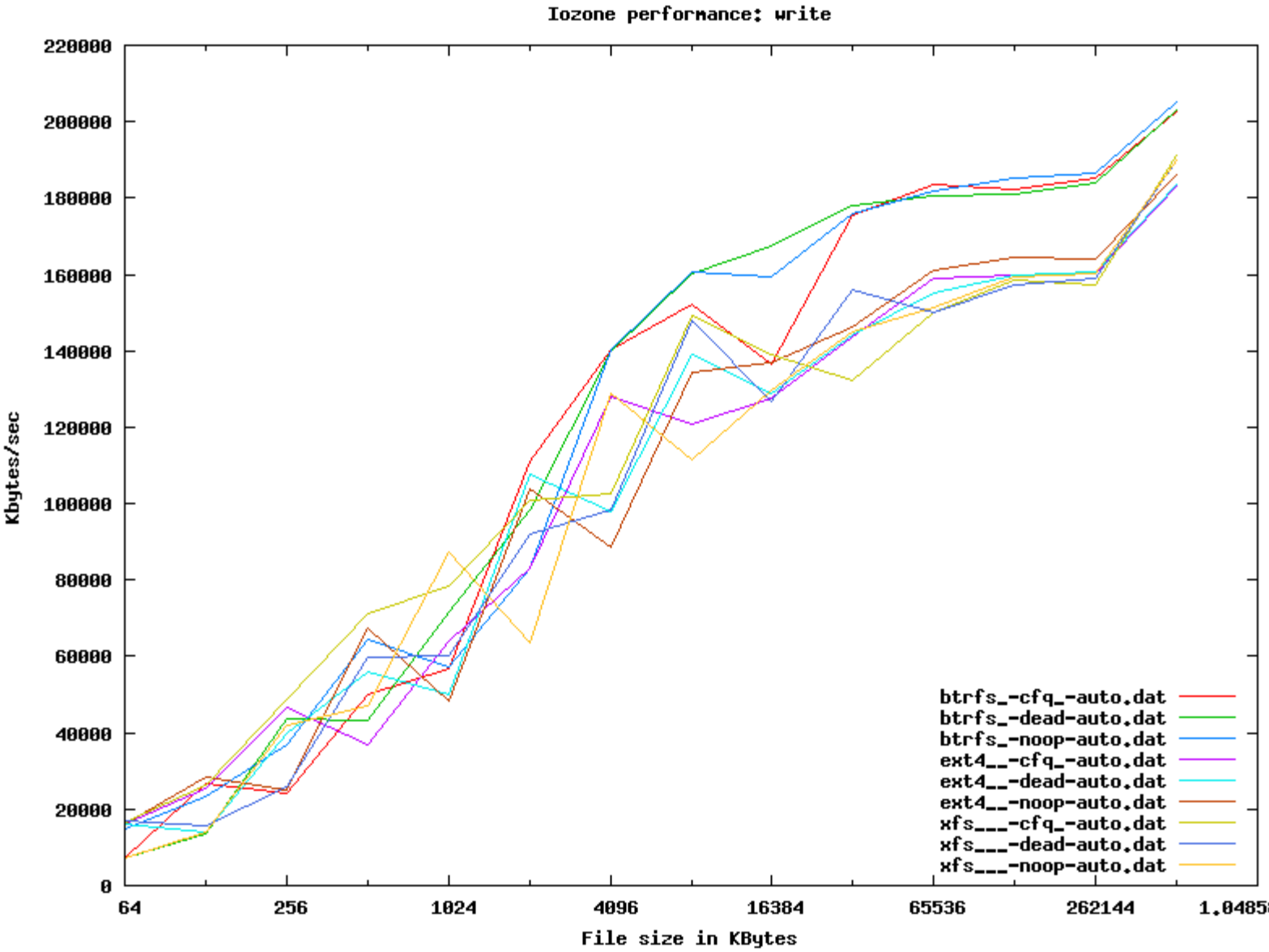
Comparing Filesystems

Storage Backend HDD² – write



Comparing Filesystems

Storage Backend SSD[1] – write



Comparing Filesystems

Storage Backend

- HDD[1] = Single HDD (Notebook)
- HDD[2] = SAS Drive (Server)
- SSD[1] = Consumer SSD (Notebook)

Not covered today

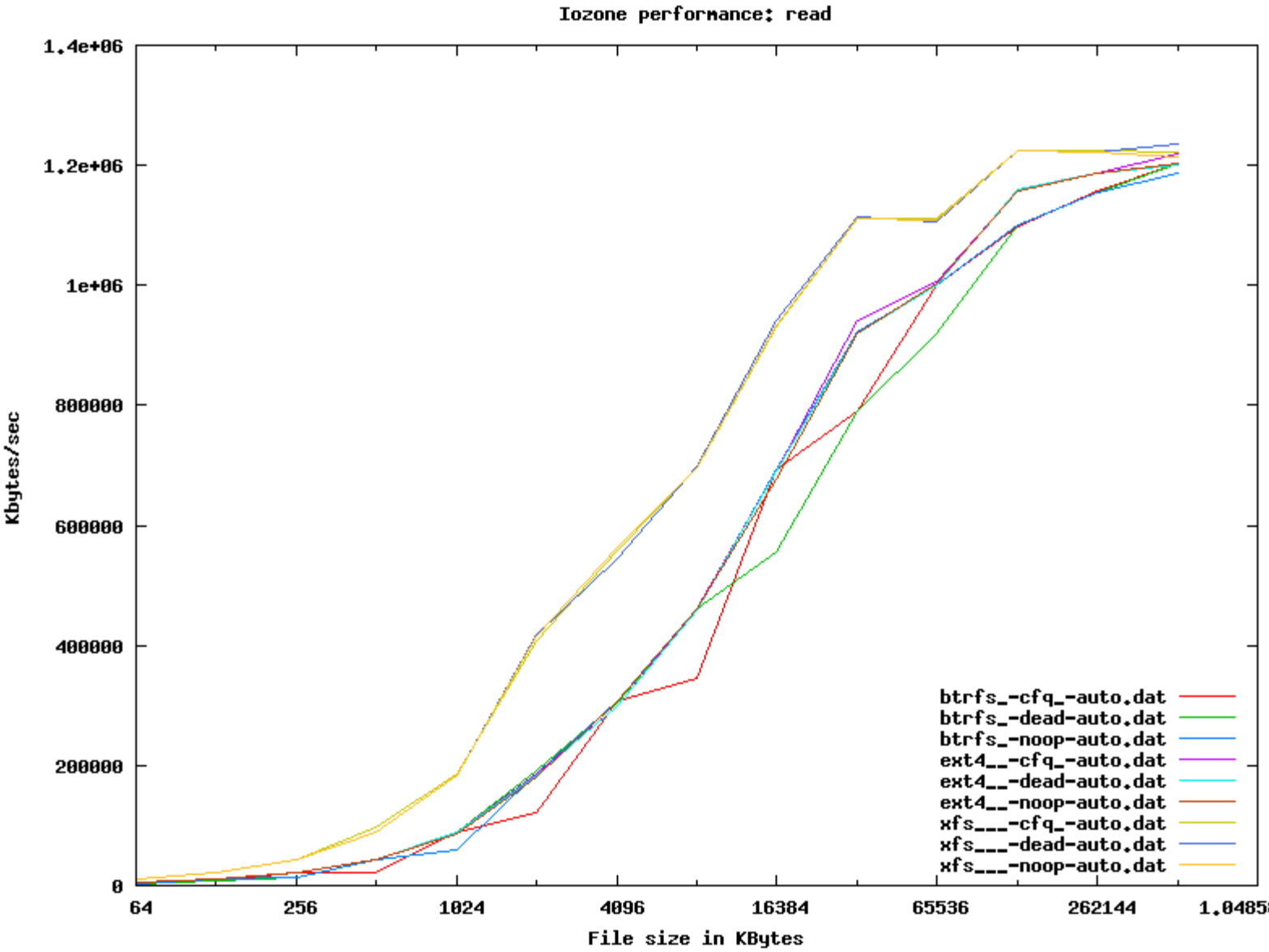
- MD RAID
- LVM
- Multipathing
- Virtual Machines / Hypervisors

The btrfs SSD optimization is visible

Workload – Read vs. Write

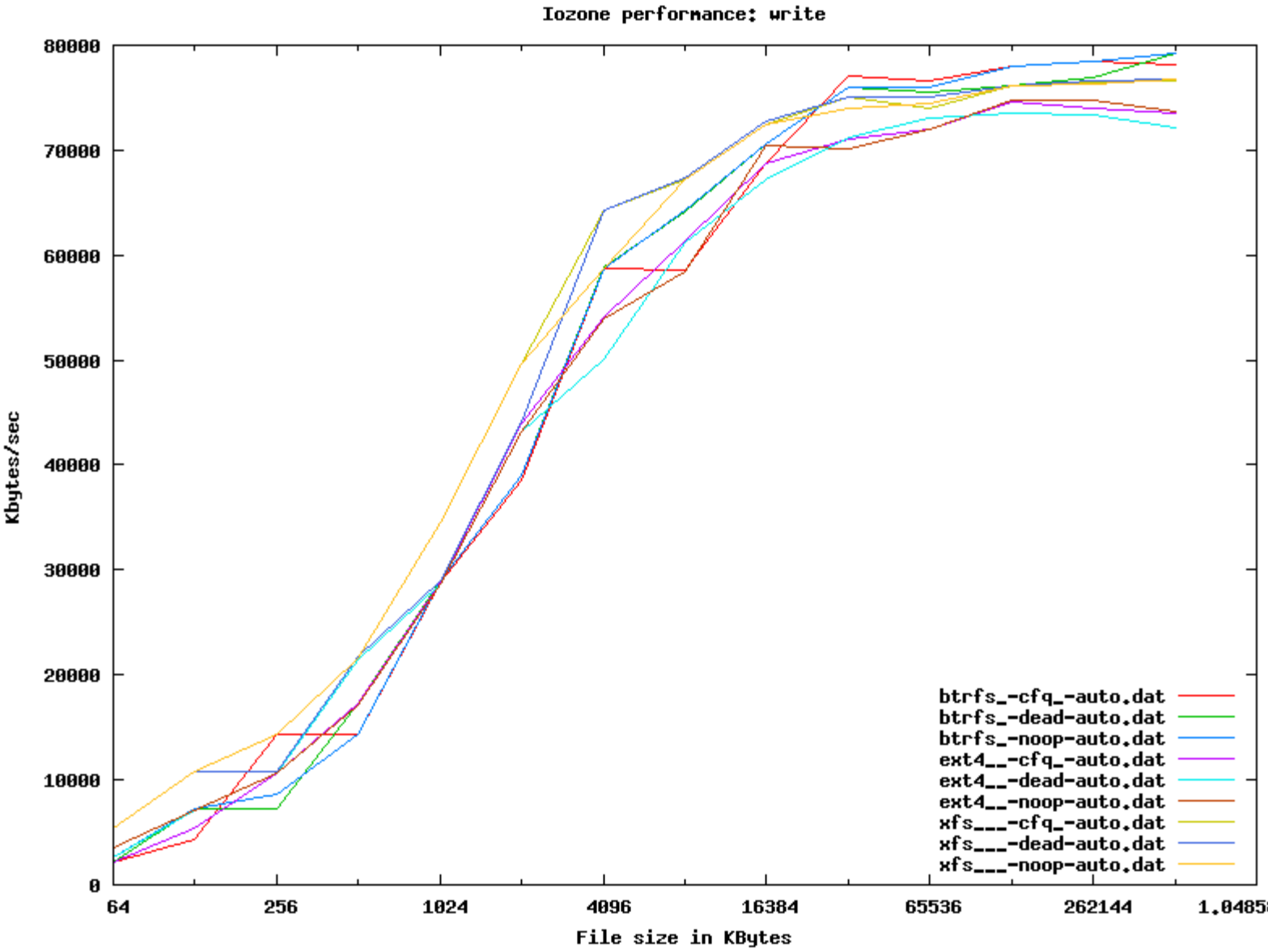
Comparing Filesystems

Storage Backend HDD² – read



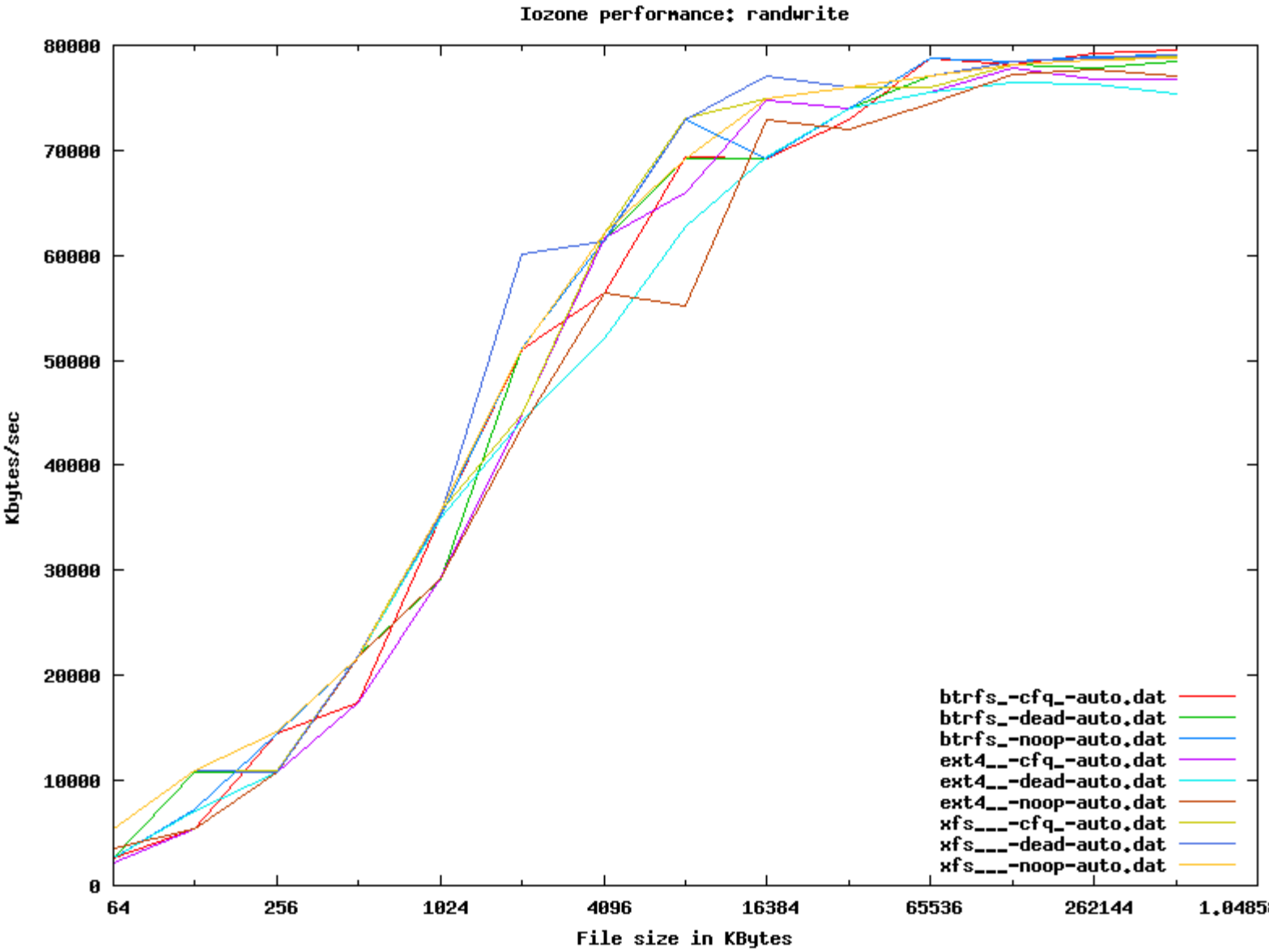
Comparing Filesystems

Storage Backend HDD² – write



Comparing Filesystems

Storage Backend HDD² – randwrite



Comparing Filesystems

read vs. write

Analyse your workload!

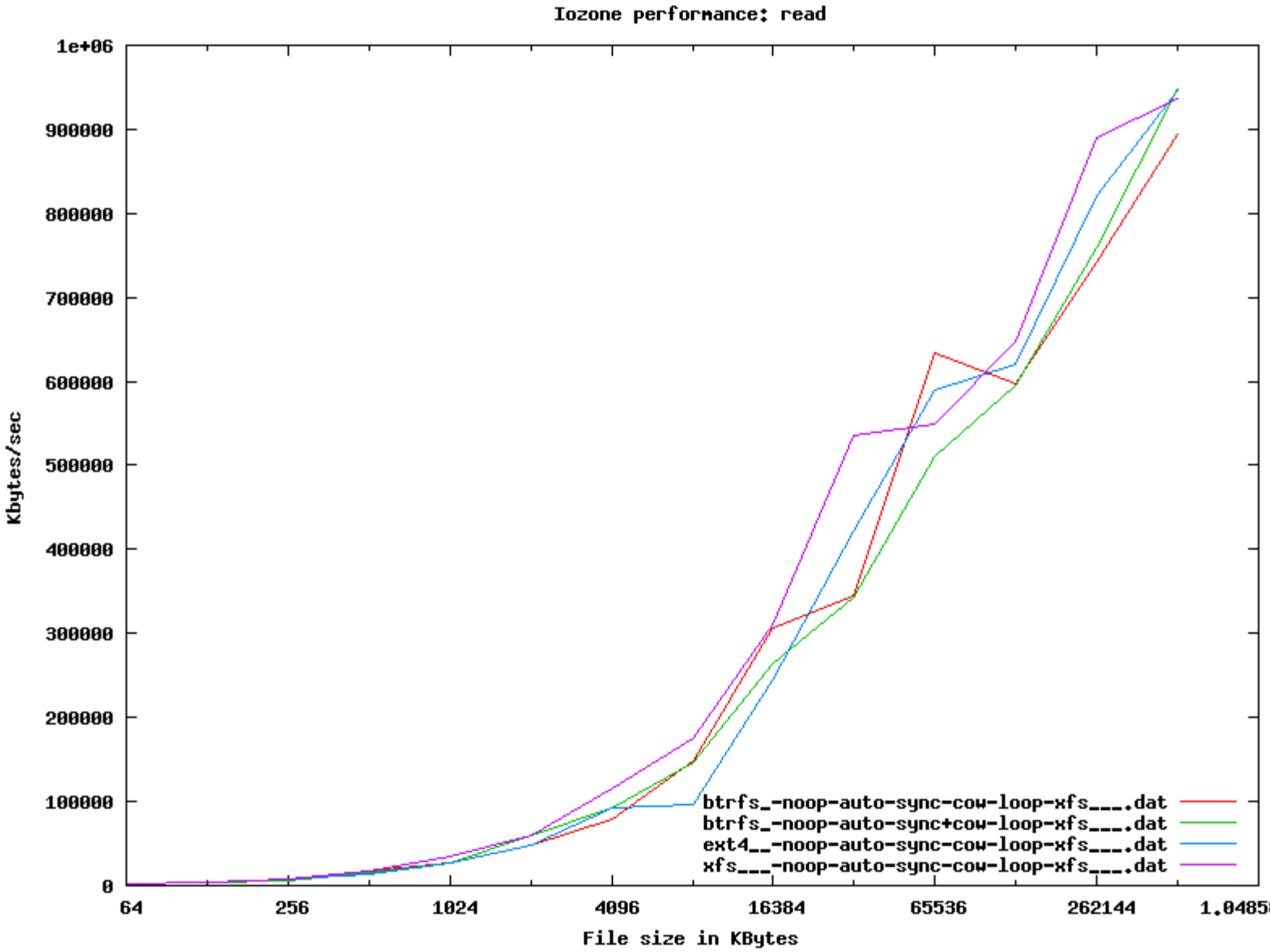
Workload – Database / VM-Host (simulated)

Or

The influence of CoW

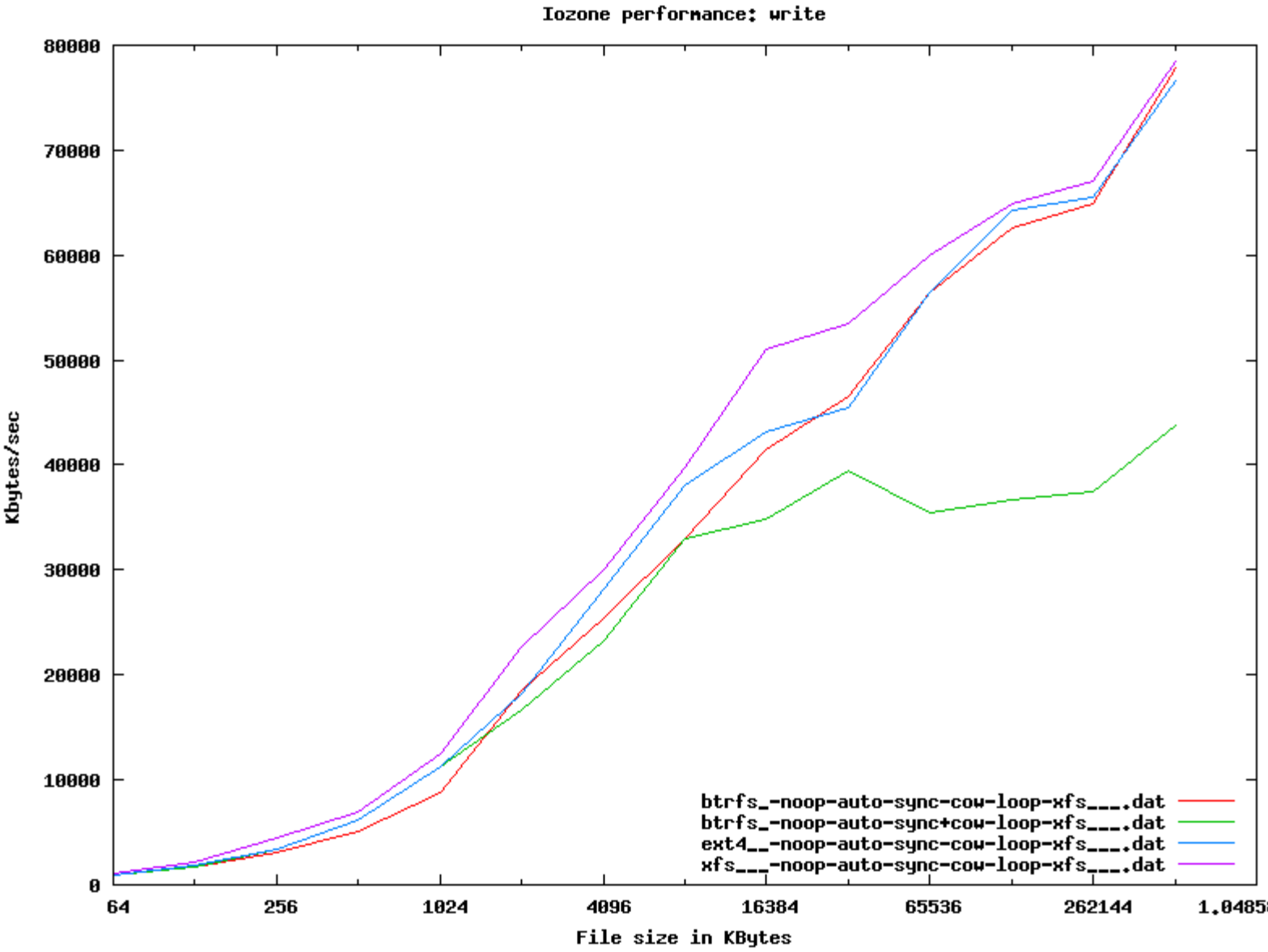
Comparing Filesystems

HDD[1] (noop, xfs as guest) – read

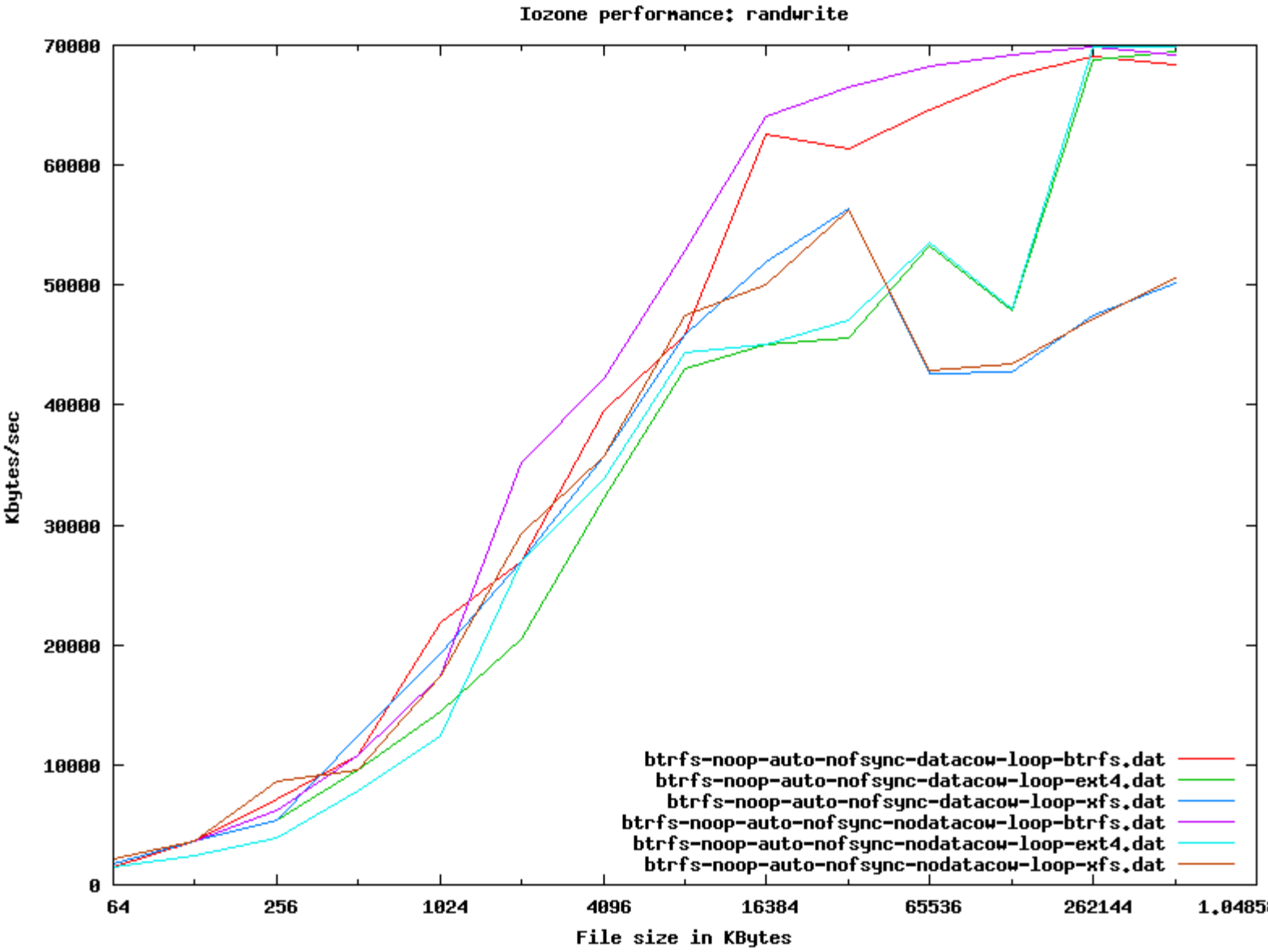


Comparing Filesystems

HDD[1] (noop, xfs as guest) – write



Comparing Filesystems
HDD[1] (noop, btrfs as host) – randwrite



Comparing Filesystems Advice

Using btrfs in a situation, where

- large files

are

- written / changed randomly,

either

- do not use btrfs (as “host”)

or

- use btrfs with “nocow” mount option

Typical use cases matching this description

- Virtual machine hosts
- Database Servers

Other Features

Major Linux (local) Filesystems

Feature	ext 2/3	reiserfs	xfs	ext4	btrfs
Data/Metadata Journaling	•/•	•/•	○/•	•/•	○
Journal internal/external	•/•	•/•	•/•	•/•	○
Copy on Write	○	○	○	○	•
Offline extend/shrink	•/•	•/•	○/○	•/•	•/•
Online extend/shrink	•/○	•/○	•/○	•/○	•/•
Inode-Allocation-Map	table	u.B*-tree	B+-tree	table	B-tree
Sparse Files	•	•	•	•	•
Tail Packing	○	•	○	○	•
Defrag	○	○	•	•	•
Deduplication	○	○	○	○	•
Compression	○	○	○	○	•
Encryption	○	○	○	•	○
ExtAttr / ACLs	•/•	•/•	•/•	•/•	•/•
Quotas	•	•	•	•	Subvol.
max. Filesystemsize	16 TiB	16 TiB	8 EiB	1 EiB	16 EiB
max. Filesize	2 TiB	1 EiB	8 EiB	1 EiB	16 EiB

Major Linux (local) Filesystems

Feature	ext 2/3	reiserfs	xfs	ext4	btrfs
Data/Metadata Journaling	•/•	•/•	○/•	•/•	○
Journal internal/external	•/•	•/•	•/•	•/•	○
Copy on Write	○	○	○	○	•
Offline extend/shrink	•/•	•/•	○/○	•/•	•/•
Online extend/shrink	•/○	•/○	•/○	•/○	•/•
Inode-Allocation-Map	table	u.B*-tree	B+-tree	table	B-tree
Sparse Files	•	•	•	•	•
Tail Packing	○	•	○	○	•
Defrag	○	○	•	•	•
Deduplication	○	○	○	○	•
Compression	○	○	○	○	•
Encryption	○	○	○	•	○
ExtAttr / ACLs	•/•	•/•	•/•	•/•	•/•
Quotas	•	•	•	•	Subvol.
max. Filesystemsize	16 TiB	16 TiB	8 EiB	1 EiB	16 EiB
max. Filesize	2 TiB	1 EiB	8 EiB	1 EiB	16 EiB

Summary

SUSE Linux Enterprise

Why ext4?

“We have ever done it this way”

- Part of the established Linux filesystem family “ext”
- Easy upgrade path from Ext2/Ext3
- Easy to learn and manage
- Built-in encryption (Kernel 4.1+)

SUSE Linux Enterprise

Why xfs?

The “Workhorse”

- Real UNIX filesystem (comes from IRIX)
- Proven reliability and petabyte scalability
- Forward and backward compatibility
- Constantly high performance under all conditions
- Basis for scale-out filesystem such as
  - Ceph

SUSE Linux Enterprise
Why btrfs?

"Copy on Write"

- CoW features
  - Snapshotting
  - Deduplication
  - Send/Receive (Future)
- Integrity features
- Management
  - Integration with Volume Management
- Built-in compression
- Basis for scale-out filesystem such as
  - Ceph

SUSE® Linux Enterprise 12

Use Cases and Filesystems

Use Case	btrfs	ext4	xfs
Need for Deduplication (Backup Server)	++	--	--
Container Host	++	+	+
Database	+[1]	+	++
Fileserver (NFS, Samba)	++	+	++
Home Directory (no Quota)	++	++	++
Home Directory (with Quota)	o[3]	++	++
Operating System	++	+	+
Need for Snapshots	++	o[2]	o[2]
VM Host	+[1]	+	++

+[1] with NoCoW

o[2] Snapshots via DM/LVM

o[3] subvolume quota only

SUSE® Linux Enterprise 12

Use Cases and Filesystems

Use Case	btrfs	ext4	xfs
Need for Deduplication (Backup Server)	++	--	--
Container Host	++	+	+
Database	+[1]	+	++
Fileserver (NFS, Samba)	++	+	++
Home Directory (no Quota)	++	++	++
Home Directory (with Quota)	o[3]	++	++
Operating System	++	+	+
Need for Snapshots	++	o[2]	o[2]
VM Host	+[1]	+	++

+[1] with NoCoW

o[2] Snapshots via DM/LVM

o[3] subvolume quota only

Comparing Filesystems

Your Filesystem!

1. Write down your workload requirements

2. Select options according to features

3. Test the options in your environment!

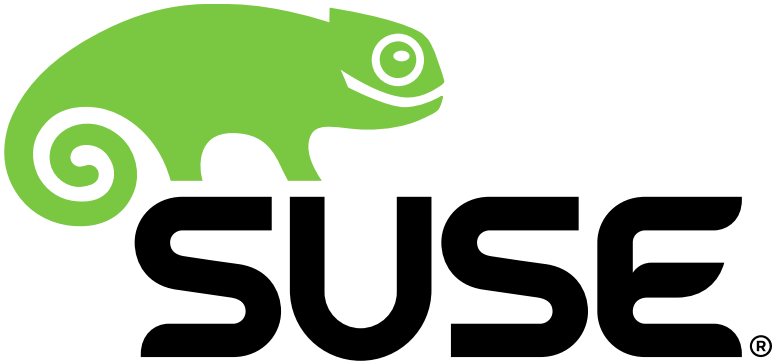
Comparing Filesystems

Your Filesystem!

Test!

Your questions?

Thank you.



Corporate Headquarters
Maxfeldstrasse 5
90409 Nuremberg
Germany

+49 911 740 53 0 (Worldwide)
www.suse.com

Join us on:
www.opensuse.org