# MySQL on ZFS for Linux

It’s not just for backups anymore

# MySQL on ZFS for Linux for Backups

Copy on write

- Pay the price in performance up front
- But backups are cheap on the backend

# More data is like rising oil prices

The tradeoff becomes more compelling as logs and data grow into the terabytes

Backups and point-in time recovery become more expensive

But what about performance?

# The price of ZFS is coming down

“We can have zfs beat ext4 on perf if we do some I/O scheduler tuning.”

Alek Pinchuk

# Start with MySQL Tuning

Given that we're using SSDs, there no need to reduce seek times

```
Innodb_flush_neighbors          = 0
```

Use checksum algorithm that scans block 32, instead of 8, bits at a time and can take advantage of optimized hardware

```
innodb_checksum_algorithm = crc32
```

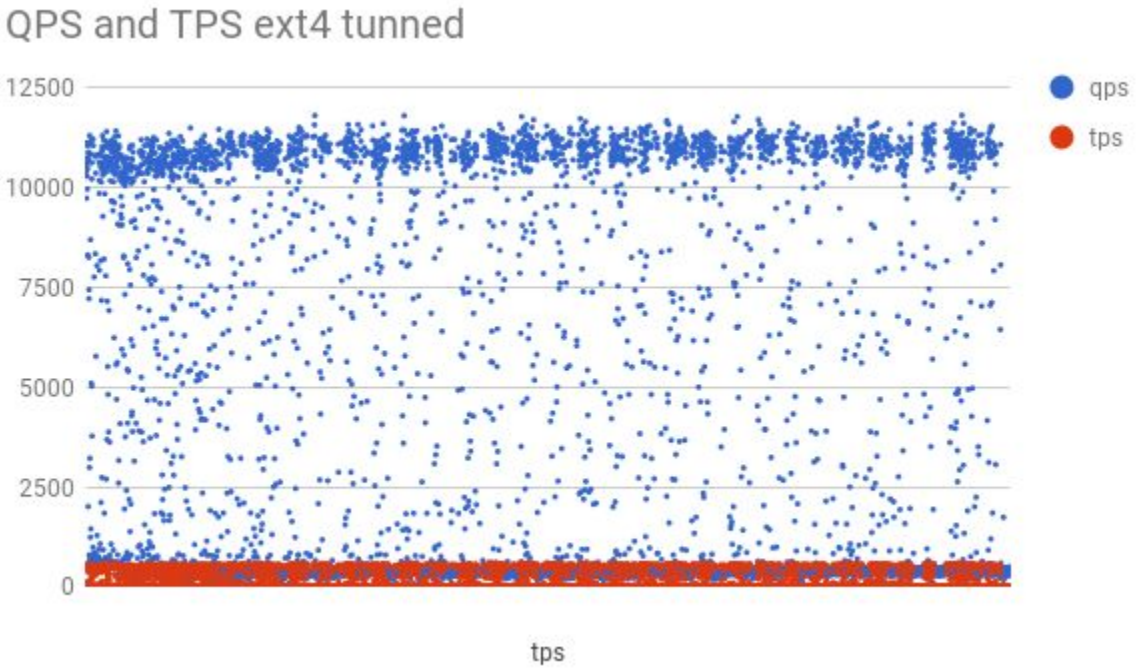
Bypass database buffering and rely on ZFS

```
Innodb_doublewrite          = 0
innodb_flush_method         = O_DSYNC
```

# Baseline ext4: mdadm RAID 1 on 120GB Kingston

```
/dev/md0:

     Raid Level : raid1

     Array Size : 105355264 (100.47 GiB 107.88 GB)

  Used Dev Size : 105355264 (100.47 GiB 107.88 GB)

   Raid Devices : 2

  Total Devices : 2

    Persistence : Superblock is persistent

  Intent Bitmap : Internal

  Spare Devices : 0
```

# Baseline: MySQL on ext4 with MySQL Tuning



# Add ZFS: 3 Mirrors on 120GB Kingston Consumer

```
mysql
  mirror-0
    ata-KINGSTON_SUV400S37120G_50026B77770488B4
    ata-KINGSTON_SUV400S37120G_50026B7777048D5D
  mirror-1
    ata-KINGSTON_SUV400S37120G_50026B7777048E50
    ata-KINGSTON_SUV400S37120G_50026B7777048E6B
  mirror-2
    ata-KINGSTON_SUV400S37120G_50026B77770488B0
    ata-KINGSTON_SUV400S37120G_50026B77770488AF
```

# MySQL on ZFS: MySQL Tuning ONLY

# ZFS Tuning: Where we started

in "/etc/modprobe.d/zfs.conf" so that they persist across reboots:

Give as much ARC as possible without starving the database buffer pool

```
options zfs zfs_arc_max=17179869184
```

ZFS prefetch not needed because MySQL InnoDB will handle that

```
options zfs zfs_prefetch_disable=1
```

Influences how often dirty data is flushed to disk; align with disk capacity to prevent stalling

```
options zfs zfs_txg_timeout=2
```

Complements timeout above; write as much data as possible as frequently as possible to minimize disk idle time

```
options zfs zfs_dirty_data_max=268435456
```

# ZFS I/O scheduler tuning: What we added

OpenZFS has separate I/O classes for read, write and scan (repair) I/O.

Each I/O class controls the min and max number of in-flight I/Os we will issue per drive

The default settings are optimized for spinning disks. Since SSDs are faster and are able to deal with more I/O at a time than HDDs:

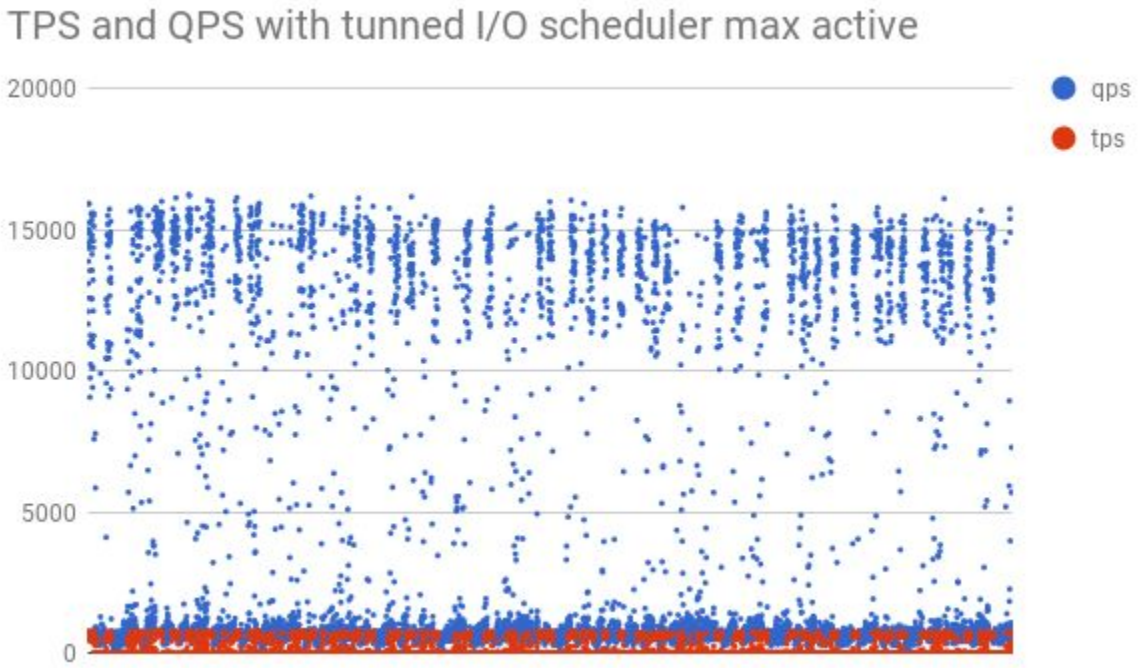
```
option zfs zfs_vdev_sync_read_max_active=32
option zfs zfs_vdev_async_read_max_active=32
option zfs zfs_vdev_sync_write_max_active=32
option zfs zfs_vdev_async_write_max_active=32
```

We increased maximum number of in-flight I/Os of each read and write class to 32

I/O scheduler tuning is hardware specific

Raising the max_active eventually causes higher latencies for other classes

# MySQL on ZFS with Tuning



# Why does this work?

We’ve increased the number of I/Os being send to disk in order to take advantage of our SSDs

# MySQL on ZFS for All Systems

Because we can achieve the required low latency throughput, we are rolling ZFS out to all our MySQL hosts.

- Fast snapshot and point in-time recovery
  - No need for rsync, xtradbbackup or mysqldump, or even binary logs
- Encryption
  - No need for LUKS

Help Wanted!

# Resources

For more info on Open ZFS tuning see:

http://dtrace.org/blogs/ahl/2014/08/31/openzfs-tuning/

https://zfs.datto.com/slides/pinchuk.pdf

For more info on MySQL on ZFS see:

https://www.percona.com/blog/2017/11/15/zfs-from-a-mysql-perspective/