ZFS and MySQL on Linux, the Sweet Spots
MySQL

The World's Most Popular Open Source Database
ZFS

Is MySQL for storage.
ZFS + MySQL

MySQL Needs
- A reliable, durable, performant storage

ZFS Provides
- A reliable, durable, performant(?) storage
ZFS + MySQL

MySQL Needs

- A reliable, durable, performant storage
- At the same time, users demand:
  - At rest encryption (number of choices available)
  - Compression (InnoDB compression is a bit complex for many)
  - Reliable and sane backups (except when you leave your dataset to grow)

ZFS Provides

- A reliable, durable, performant(?) storage
ZFS + MySQL

MySQL Needs

- A reliable, durable, performant storage
- At the same time, users demand:
  - At rest encryption (number of choices available)
  - Compression (InnoDB compression is a bit complex for many)
  - Reliable and sane backups (except when you leave your dataset to grow)

ZFS Provides

- A reliable, durable, performant(?) storage
- Compression options, encryption, sane backups
ZFS + MySQL

It's a compromise and they should meet somewhere in between ...
WARNING: You will see graphs ...
Use Case: Large MySQL Dataset
Large MySQL Dataset

Probably not what you should be doing, but:

- Single large dataset, in the TBs range
  - Backup and recovery challenges
  - Storage space constraints
  - Long data retention periods (fintech/healthcare)
- Mixed storage engines (yes MySQL allows it)
Switching is not as straightforward however:

- Thick performance bands on ZFS
- TPS down to 0 on write heavy tests
Large MySQL Dataset

Switching is not as straightforward however:

- Gets worse on 32 threads, write-only workload
Switching is not as straightforward however:

- Transactions/sec drops to zero a lot of times!
Switching is not as straightforward however:

- HP DL380G7 6xSAS disks, P410 (with battery backed cache)
- Disks configured as JBOD, ZFS pool 2x3 mirrored stripes

### ZFS

- `logbias=throughput`
- `zfs_arc_max=1073741824`
- `zfs_prefetch_disable=1`
- `atime=12`
- `recordsize=16k/128k`

### MySQL

- `sync_binlog=1`
- `innodb_flush_log_at_trx_commit=1`
- `innodb_doublewrite=0`
Trying to optimize without shelling out extra cash:

Minimum TPS drops just by tuning `zfs_dirty_data_max=128M`:

```
./1-no-slog/sb-no-slog-ui32.csv  383
./7-no-slog-no4-txgtimeout5/sb-no-slog-no4-txgtimeout5-ui32.csv  18
```
Large MySQL Dataset

Trying to optimize without shelling out extra cash:

- Still quite far off from EXT4 numbers, but we know our ceiling
Biting the bullet, adding an NVMe SLOG:

- Raised further, but not as far as we expected
- `zfs_nocacheflush=1` helps too, we are using RAID controller with battery backed cache
Biting the bullet, adding an NVMe SLOG:

- Limited by how much throughput and IOPs combined our main pool disks can deliver.
Well let's test with SSD as well:

- Same hardware, 6x Samsung 860 SSD drives
- Same ZFS config without NVMe SLOG
- Big gap from EXT4, just a matter of tuning for capacity with SSD
Large MySQL Dataset

Unfortunately, just as we were having fun:

```
¯\_(ツ)_/¯
```

Watch this space:
https://github.com/dotmanila/zfs-mysql
What we learned so far:

- Know your pool capability and capacity
- Performance depends on the slowest component (pool disks vs slog)
Use Case: Percona XtraDB Cluster
Percona XtraDB Cluster

A group of MySQL servers with:

- Synchronous Replication
- Multi Master*, True Parallel Replication
- Automatic node provisioning

*Requires compatible workload type
How Galera replication works:

Percona XtraDB Cluster

Why ZFS fits:

- Writeset certifications within the cluster allows least per node durability.
  - `sync_binlog = 0`
  - `innodb_flush_log_at_trx_commit = 0`
Why ZFS fits:

- We should be able to tune ZFS with least durability as well.
  - sync=disabled
  - zfs_nocacheflush=1
- Plus PXC settings:
  - innodb_doublewrite=0
  - innodb_log_checksums=OFF
  - innodb_checksum_algorithm=none
Exploring ZFS on PXC, i3.2xlarge
Exploring ZFS on PXC, i3.2xlarge

```
[mysqld]
server-id=1
datadir=/mysql/data
wsrep_provider=/usr/lib/galera3/libgalera_smm.so
wsrep_cluster_address=gcomm://
binlog_format=ROW
default_storage_engine=InnoDB
innodb_doublewrite=0
innodb_log_group_home_dir=/mysql/logs
innodb_io_capacity=5000
innodb_autoinc_lock_mode=2
innodb_flush_log_at_trx_commit=0
innodb_buffer_pool_size=48G
innodb_log_file_size=8G
innodb_log_checkums=OFF
innodb_checksum_algorithm=none
pxc_strict_mode=ENFORCING
wsrep_node_name=zfs01
wsrep_slave_threads=8
wsrep_node_address=10.1.2.117
wsrep_cluster_name=zfs
wsrep_sst_method=xtrabackup-v2
wsrep_sst_auth="msandbox:msandbox"
```

```
options zfs zfs_arc_max=1073741824
options zfs zfs_prefetchDisable=1
options zfs zfs_nocacheflush=1

sudo zpool create -o ashift=12 -f mysql /dev/nvme0n1
sudo zfs set recordsize=16k mysql
sudo zfs set atime=off mysql
sudo zfs set logbias=latency mysql
sudo zfs set primarycache=metadata mysql
sudo zfs set compression=lz4 mysql
sudo zfs set sync=disabled mysql

sudo zfs create -o recordsize=128K mysql/logs
sudo zfs create -o recordsize=16K mysql/data
```
Percona XtraDB Cluster

Why ZFS fits:

- And still take advantage of ZFS features:
  - ZFS snapshots for SST
    - Reading records/blocks from source multithreaded
    - Writing records to destination is also multithreaded
  - Encryption
  - Compression
State Snapshot Transfer:

- **XtraBackup**
  - Donor becomes unavailable, on a 3 node cluster, you lose 2/3 when one needs SST.
  - Potentially slow for large uncompressed datasets
    - On the fly compression and decompression
    - It is however parallel
    - Depends on compressability

- **ZFS Snapshot**
  - Donor remains available
  - Raw SEND/REC
State Snapshot Transfer:

- Taking ZFS Snapshot with Percona Server/Percona XtraDB Cluster

```sql
mysql> LOCK TABLES FOR BACKUP;
mysql> LOCK BINLOG;
mysql> SHOW MASTER STATUS;
mysql> -- take ZFS snapshot here
mysql> UNLOCK TABLES;
mysql> UNLOCK BINLOG;
```
State Snapshot Transfer:

- Streaming encrypted and compressed:

```bash
ubuntu@ip-10-1-2-117:~$ time sudo zfs send -wcR mysql@201804192210 | mbuffer -s 128
in @ 22.2 MiB/s, out @ 76.4 MiB/s, 156 GiB total, buffer 0% full
summary: 156 GiByte in 23min 22.1sec - average of 114 MiB/s

real 23m24.961s
user 0m9.052s
sys 6m39.036s
```

```bash
ubuntu@ip-10-1-2-126:~$ mbuffer -s 128k -m 1G -I 9999 | sudo zfs recv -F mysql
in @ 77.9 MiB/s, out @ 77.9 MiB/s, 156 GiB total, buffer 0% full
summary: 156 GiByte in 23min 24.7sec - average of 114 MiB/s
```

```bash
ubuntu@ip-10-1-2-126:~$ sudo zfs load-key -a
2 / 2 key(s) successfully loaded
```

```bash
ubuntu@ip-10-1-2-126:~$ sudo zfs mount -a
```
Percona XtraDB Cluster

State Snapshot Transfer: ZFS POC Tests

- r4.4xlarge, single EBS 4000 PIOPs
- PXC 5.7.19
- Idle cluster, no traffic during snapshot transfers

<table>
<thead>
<tr>
<th>Config</th>
<th>Event</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ext4</td>
<td>prepare</td>
<td>21m31.831s</td>
</tr>
<tr>
<td>sst</td>
<td></td>
<td>3m5s</td>
</tr>
<tr>
<td>sst (threads=8)</td>
<td></td>
<td>3m5s</td>
</tr>
<tr>
<td>zfs (xenial)</td>
<td>prepare</td>
<td>40m53.066s</td>
</tr>
<tr>
<td>sst</td>
<td></td>
<td>9m37s (same for 8thd)</td>
</tr>
<tr>
<td>send/recv</td>
<td></td>
<td>10min 48.9sec</td>
</tr>
<tr>
<td>send -c/recv</td>
<td></td>
<td>NA</td>
</tr>
<tr>
<td>zfs (git)</td>
<td>prepare</td>
<td>33m56.747s</td>
</tr>
<tr>
<td>sst</td>
<td></td>
<td>4m35s</td>
</tr>
<tr>
<td>sst (threads=8)</td>
<td></td>
<td>3m22s</td>
</tr>
<tr>
<td>send/recv</td>
<td></td>
<td>4min 03.6sec</td>
</tr>
<tr>
<td>send -c/recv</td>
<td></td>
<td>4min 02.3sec</td>
</tr>
<tr>
<td>zfs (git, gzip)</td>
<td>prepare</td>
<td>45m23.769s</td>
</tr>
<tr>
<td>sst</td>
<td></td>
<td>3m40s</td>
</tr>
<tr>
<td>send/recv</td>
<td></td>
<td>6min 39.0sec</td>
</tr>
<tr>
<td>send -c/recv</td>
<td></td>
<td>2min 55.3sec</td>
</tr>
</tbody>
</table>
Use Case: MySQL Dedicated Backup Nodes
MySQL Dedicated Backup Nodes

Backups are fun:

- Performance
- Restore Time Objective
- Restore Point Objective
- Security
Backup options for InnoDB:

- Percona XtraBackup
- Delayed Replica
MySQL Dedicated Backup Nodes

Backups are fun: XtraBackup (1/2)

- Performance
  - Fast parallel copy, fast on the fly compression with qpress
  - Decompress in parallel too

- Restore Time Objective
  - Depends on where backups are stored relative to restore target
  - How fast to transfer backups from stored state to usable state
  - PITR also adds up and depends on how frequent backups are made
  - Full + incremental backups are possible
MySQL Dedicated Backup Nodes

Backups are fun: XtraBackup (2/2)

- Restore Point Objective
  - PITR with full + incremental + binary logs roll forward
  - Gap between incremental and binary log target is crucial

- Security
  - Encryption on the fly, decryption separate process
  - Decompression and decryption can be done at the same time in parallel with some shell-fu
  - `cat table.ibd.xbcrypt.qp | xbcrypt -d | qpress -d | table.ibd`
MySQL Dedicated Backup Nodes

Backups are fun: Delayed MySQL Replica (1/2)

- Performance
  - Full dataset immediately available
  - PITR recovery speed depends on configured delay and SQL thread speed

- Restore Time Objective
  - Depends on configured delay and SQL thread speed
MySQL Dedicated Backup Nodes

Backups are fun: Delayed MySQL Replica (2/2)

- Restore Point Objective
  - PITR range depends on configured delay (1hr vs 1day)
- Security
  - Scoped within instance security (at rest, in transit)
MySQL Dedicated Backup Nodes

Testing Snapshots

- Run sysbench oltp_update_index test, 32 threads on an i3.2xlarge
- Take snapshots every 5mins, copy the last snapshot every hour (send to /dev/null)

```bash
while true; do
    echo "$(date +%Y-%m-%d_%H:%M) sleeping ...");
    sleep 300;
    sudo zfs snapshot -r mysql@$(date +%Y%m%d%H%M);
done

while true; do
    snap=$(zfs list -t snap | egrep 'mysql@' | tail -n1 | awk '{print $1}');
    time sudo zfs send -R $snap | cat - > /dev/null;
    sleep 3600;
    done
```
MySQL Dedicated Backup Nodes

Testing Snapshots

- Transactions/sec looks to be even and non-degrading
MySQL Dedicated Backup Nodes

Testing Snapshots

- Running the graph on Loess smoothing, reveals something interesting
- Transactions/sec degrades overtime
MySQL Dedicated Backup Nodes

Testing Snapshots

- Reads saturating the disk
  - Binary logs size increased
MySQL Dedicated Backup Nodes

Backups are fun: ZFS Snapshots (1/2)

- Performance
  - Compressed optimized SEND/RECV
- Restore Time Objective
  - Same as Xtrabackup (full + incremental + binlogs speed)
  - In place snapshots allows near instantaneous rollback!
Backups are fun: ZFS Snapshots (2/2)

- **Restore Point Objective**
  - In place snapshots allows rollback as far as space allows
  - Combined with replication, essentially also a delayed replica but faster!

- **Security**
  - Encrypted datasets can remain encrypted in transit and to destination
Looking Forward
Looking Forward

Exciting things up ahead:

- ZSTD compression!
- More MySQL related potential tuning:
  - Separate undo log directory - InnoDB undo log are best suited for SSDs
- PXC state snapshot transfer (automatic provisioning) with ZFS snapshots
Questions!