Multi-Namespace Management

& Performance Optimization

Ron Yuan
Vice President of Firmware
Customers Need Simple Adoption of New NVMe Features

1. Utilization of Multiple Namespace and Quota by Namespace

2. Solve MySQL Doublewrite Bottleneck and Over-consumption on SSD Endurance with Mixed-media based Multi-Namespace Management
Flexible Utilization with Multiple Namespace

Benefits of Multi-Namespace for SSD with high capacity:
- Lower cost per GB
- Space saving
- Multiple users/applications

For example:
SSD supports only 1 namespace

```
# nvme list
/dev/nvme0n1 -- 8TB
```

PBlaze5 8TB U.2
Create 4 namespaces

```
# nvme list
/dev/nvme0n1 -- 1TB
/dev/nvme0n2 -- 1TB
/dev/nvme0n3 -- 3TB
/dev/nvme0n4 -- 3TB
```

Multi-Namespace on PBlaze5:
- PBlaze5 SSD supports up to 32 namespaces
- Standard management command (nvme create-ns)
- Different AES-256 key
- Different sector size / PI
- Share capacity and performance
- A big problem for customer: when two applications share the same SSD, how can the SSD evenly serve two of them?

- For most of SSDs, greedy application gets more service, slower application needs to suffer long latency

- IOD serve the needs but also brings problems

Noisy Neighbor (NS2) Effect: NS1 Latency Increment

Performance is measured @4K Rand Read 50K IOPS
PBlaze5 QoS Improvement with Quota by Namespace

- Memblaze’s solution is to provide customer with VS command to set Bandwidth Quota for each namespace.

- Easy setup, flexible to use.

- Example:
  Create 8 namespaces with the same size
  NS1~4 Seq Read: 391MB/s
  NS5~8 Seq Read: 39MB/s
Customers need simple adoption of new NVMe features

1. Utilization of Multiple Namespace and Quota by Namespace

2. Solve MySQL Doublewrite Bottleneck and Over-consumption on SSD Endurance with Mixed-media based Multi-Namespace Management
MySQL Doublewrite Buffer & Doublewrite Space

Doublewrite is a mechanism to prevent data corruption during accident power loss, partial data is written to the drive.

Doublewrite Space:
- Data is written twice, in some heavy workloads the doublewrite buffer becomes a performance bottleneck
- Massive writes lead to SSD wears out quickly
Ibdata1 including:
1. Data dictionary
2. Doublewrite space
3. Insert buffer
4. Rollback segments
5. UNDO space
6. Foreign key constraint
7. System tables

Normal Solution:
Put doublewrite buffer on separated drive using high performance media (MRAM/PCM/Xpoint), isolates with NAND based SSD.

Normal Solution 2:
Use Atomic write feature to replace double write buffer.

Memblaze Solution:
Put doublewrite buffer on DRAM based namespace,
Doublewrite Buffer Analyze

1. Double write buffer (DWB) is very small in size
2. only used after sudden power loss event.
3. Enterprise SSD has large DDR to serve as DWB
4. Enterprise SSD has native power loss protection by capacitor.

Here we use Percona MySQL as test case:
1. Percona Parallel Doublewrite Buffer is designed to solve the performance bottleneck which is introduces by traditional Doublewrite buffer.
2. Percona doublewrite space has been separated into a single file (non-tablespace). This file contains shards for all buffer pool instances. Each shard has different offsets.

4 buffer pool instances allocate 8 doublewrite shards, nearly 16MB
MySQL TPCC Test Environments

1. PowerEdge R730xd
   (1) CPU: Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz 8 Cores * 2
   (2) Memory: DDR4 96GB
   (3) Memblaze PBlaze5 910 NVMe SSDs:
      3.84TB U.2 SSD with 3.84TB namespace * 1 (nvme0n1)
      3.84TB U.2 SSD with 3.80TB namespace * 1 (nvme3n1) and 64MB DRAM Namespace *1 (nvme3n2)

2. Centos 7.4 with NVMe driver 1.0, ext4 filesystem
3. Percona MySQL 8.0.15
datadir=/data1, innodb_doublewrite=on, innodb_parallel_doublewrite_path=/DWB/doublewrite.file
VS.
datadir=/data1, innodb_doublewrite=on, innodb_parallel_doublewrite_path=/data1/xb_doublewrite
innodb_buffer_pool_size=16GB, innodb_buffer_pool_instances=8 => need 32MB double write file
innodb_flush_log_at_trx_commit = 1, innodb_flush_method=O_DIRECT
innodb_write_io_threads=16, innodb_read_io_threads=8
innodb_io_capacity=10000, log_bin=/data1/mysql-bin

4. TPCC MySQL
   (1) connections=4,8,16,32,64,128
   (2) warehouse=28000 => Test data amount is 3TB
   (3) warmup_time=600
   (4) running_time=10800

Use Memblaze customized firmware, customer can allocate Namespace from DDR space like a RAM disk.
MySQL TPCC Test Results

Put InnoDB parallel double write file on DRAM Namespace, performance **improves 35.49%** under 64 thread concurrency.

<table>
<thead>
<tr>
<th>THREAD</th>
<th>TpmC_1: MySQL Data at nvme0n1, double write buffer at nvme0n1</th>
<th>TpmC_2: MySQL Data at nvme3n1, double write buffer at nvme3n2(64MB DRAM NS)</th>
<th>TpmC_2-TpmC_1</th>
<th>(TpmC_2-TpmC_1)/TpmC_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>9505.272</td>
<td>10814.7</td>
<td>1309.428</td>
<td>13.78%</td>
</tr>
<tr>
<td>8</td>
<td>15678.145</td>
<td>18124.768</td>
<td>2446.623</td>
<td>15.61%</td>
</tr>
<tr>
<td>16</td>
<td>23205.877</td>
<td>27210.906</td>
<td>4005.029</td>
<td>17.26%</td>
</tr>
<tr>
<td>32</td>
<td>29988.883</td>
<td>39297.746</td>
<td>9308.863</td>
<td>31.04%</td>
</tr>
<tr>
<td>64</td>
<td>31138.277</td>
<td>42189.027</td>
<td>11050.75</td>
<td>35.49%</td>
</tr>
</tbody>
</table>
MySQL TPCC Test Results -- IO Press

Put InnoDB parallel double write file on DRAM Namespace:
1. SSD Avg Random Read IOPS and Random Read throughput improves 38.3%;
2. SSD Avg Random Write throughput improves 37.1%;
3. SSD Avg Random Read latency reduces 7.4%;
4. SSD Avg Random Write latency reduces 52.6%;

Percona_8.0.15 TPCC TEST Parameters: Warehouse=28000 Warmup_time=600 Running_time=10800
MySQL Parameters: innodb_buffer_pool_size=16GB innodb_buffer_pool_instances=8 innodb_flush_log_at_trx_commit=1

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TpmC_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nvme0n1(MySQL Data &amp; double write buffer)</td>
<td>46888.194</td>
<td>732.626</td>
<td>16</td>
<td>20410.696</td>
<td>540.13</td>
<td>27.1</td>
<td>0.405</td>
<td>2.758</td>
</tr>
<tr>
<td>TpmC_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nvme3n1(MySQL Data)</td>
<td>64845.669</td>
<td>1013.213</td>
<td>16</td>
<td>28242.749</td>
<td>396.073</td>
<td>14.36</td>
<td>0.375</td>
<td>1.306</td>
</tr>
<tr>
<td>nvme3n2(double write buffer in 64MB DRAM NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nvme3n1 + nvme3n2</td>
<td>1013.213</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Percent

Improves 38.3%
Improves 38.3%
Improves 37.1%
Reduces 7.4%
Reduces 52.6%(MySQL Data)
## MySQL TPCC Host & NAND Write

Put InnoDB parallel double write file on DRAM Namespace, **WA reduces 46%**.

### Test Case

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Host Write(GB)</th>
<th>NAND Write(GB)</th>
<th>WA</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL Data at nvme0n1, double write buffer at nvme0n1</td>
<td>29,953</td>
<td>74,229</td>
<td>2.478</td>
</tr>
<tr>
<td>MySQL Data at nvme3n1, double write buffer at nvme3n2(64MB DRAM NS)</td>
<td>39,114</td>
<td>52,220</td>
<td>1.335</td>
</tr>
</tbody>
</table>

Percent

- **Implements 31%**
- **Reduces 30%**
- **Reduces 46%**
NVMe namespace feature provides a great possibility to manage different media, different performance, different security and more.

Customer is always asking for
- Lower cost
- Robust product
- Simplified adoption

We have seen slow adoption of NVMe new features such as streams or IOD.

Focus on what customer really needs