<table>
<thead>
<tr>
<th>저자</th>
<th>Trong-Dat Nguyen, Sang-Won Lee</th>
</tr>
</thead>
<tbody>
<tr>
<td>출처</td>
<td>한국정보과학회 학술발표논문집, 2017.06, 183-185 (3 pages)</td>
</tr>
<tr>
<td>발행처</td>
<td>한국정보과학회</td>
</tr>
<tr>
<td>Publisher</td>
<td>KOREA INFORMATION SCIENCE SOCIETY</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://www.dbpia.co.kr/Article/NODE07207175">http://www.dbpia.co.kr/Article/NODE07207175</a></td>
</tr>
<tr>
<td>이용정보</td>
<td>성균관대학교 자연과학캠퍼스</td>
</tr>
<tr>
<td>Accessed</td>
<td>115.***.173.200</td>
</tr>
<tr>
<td></td>
<td>2018/12/12 20:07 (KST)</td>
</tr>
</tbody>
</table>

Copyright Information

Copyright of all literary works provided by DBpia belongs to the copyright holder(s) and Nurimedia does not guarantee contents of the literary work or assume responsibility for the same. In addition, the literary works provided by DBpia may only be used by the users affiliated to the institutions which executed a subscription agreement with DBpia or the individual purchasers of the literary works for non-commercial purposes. Therefore, any person who illegally uses the literary works provided by DBpia by means of reproduction or transmission shall assume civil and criminal responsibility according to applicable laws and regulations.
MongoDB Journaling Evaluation with NVDIMM Devices
Trong-Dat Nguyen  Sang-Won Lee
College of Information and Communication Engineering, Sungkyunkwan University
datnguyen@skku.edu, swlee@skku.edu

Abstract
This paper evaluates and analyzes the effect of a real low-latency high-performance NVDIMM device when used as the journaling device in MongoDB, using YCSB benchmark. For comprehensive analysis, we use diverse configurations for both database and storage layers and also with both stand-alone server and replica sets. The experimental results with the NVMDIMM device have dramatically improved the throughput improvement by up to 2.5 times over the original approach.

1. Introduction
Traditional relational Database Management System (RDBMS) e.g. MySQL, PostgreSQL are popular and well developed in the last decades. However, in current big data era, wearable devices, sensors, smart phones, new generation applications and users generate data that have large-volume, heterogeneity formats, and quickly change. Such kind of data and applications requires the database system has ability to scale-out in distributed environment and flexible data model. Traditional RDBMS is inadequate to handling both requirements, therefore, next-generation NoSQL solution i.e. document stores are introduced to solve the problems.

In transaction processing, while RDBMS use ACID (Atomic, Consistency, Isolation, and Durability), NoSQL DBMS prefers BASE [1] properties (Basic availability, Soft-state, and Eventually consistent) that is more relaxing constraints compare to the traditional. One of important advantage of MongoDB is supporting transaction processing by adapting WiredTiger storage engine.

Flash Solid State Disks (SSDs) show many advantages compare to hard disks (HDDs) [2]. Moreover, Non-Volatile DIMMs (NVDIMMs) is the next generation technique that is fast as DRAM and persistent as SSD/HDD, therefore, those devices have slow data access rates and access data in block address that can lead to high write amplification. In the other hand, read/write on DRAMs is extremely rapid with byte address access, however, data are lost when power is lost and DRAM have limited capacity. Non-Volatile DIMM is introduced to exploit advantages from block devices and DRAM that is consistent from power lost as SSD/HDD and fast as DRAM, thus it necessary to modify the components in DBMSs to exploit advantages of NVMDIMMs [4,5].

2. Background
2.1. Non-Volatile DIMM (NVDIMM)
For the sake of consistency, DBMSs write data to non-volatile devices i.e. SSDs or HDDs, however, those devices have slow data access rates and access data in block address that can lead to high write amplification. In the other hand, read/write on DRAMs is extremely rapid with byte address access, however, data are lost when power is lost and DRAM have limited capacity. Non-Volatile DIMM is introduced to exploit advantages from block devices and DRAM that is consistent from power lost as SSD/HDD and fast as DRAM, thus it necessary to modify the components in DBMSs to exploit advantages of NVMDIMMs [4,5].

2.2. MongoDB Architecture
There is a logical mapping between MongoDB data model and traditional RDBMS data model. Database in MongoDB is similar with RDBMS that includes system data and user data. Collection in MongoDB is mapped with table in RDBMS. Due to fixed schema, a table in RDBMS consists of rows (records) that have the same number of columns (data fields). In the other hand, a collection in MongoDB is the set of documents which is represented by JSON format, each document consists of objects i.e. key-value pairs such that key is a string and value is atomic value, string or nested object.

For very large data set, MongoDB scales-out the
system such that data are distributed across nodes (shards) in a share-nothing environment as illustrated in Figure 1. For the sake of consistency and availability, each shard in turn is a replica set following a master-slave architecture that consists of one primary node and many secondary nodes. While a client can read data from both the primary node and secondary nodes, update requests only write on the primary node, then secondary nodes synchronize data from the primary node through OpLog records.

![Figure 1: MongoDB clustering with sharding and replica set](image)

2.3. MongoDB Journaling and write concern behaviors

MongoDB supports various options of write concern i.e. the level of acknowledgment requested from the client to a standalone server or to replica sets in distributed environment. The typical write concern request from the client is in the form of a JSON document as `{w:<number>, j:<boolean>}`. Parameter `w` means the request is acknowledged after the number of `w` mongod instances have updated in-memory data from write operations. If `w` is set to 0, there is no acknowledgment but network errors may be reported. If parameter `j` is set to true, the write request is acknowledged only after the number of mongod instances (set by `w`) have persist log records on the journal files. For example, in Figure 2, when a write operation issued from the client to a replica set with one primary node and two secondary nodes with write concern `{w:3, j:false}`, first, write operations apply on the primary, when updating data in-memory is finished, the write is synced to secondary nodes. Whenever, an in-memory data update is done in a secondary node, it acknowledges the primary. The primary waits until both secondary nodes acknowledge before responding to the client. Note that, due to `j` is set to false, each node does not wait for log records persist in journal file before acknowledging.

![Figure 2: Write concern behavior in replica set](image)

3. Evaluation

3.1. Experiment setup

We evaluate the system in both standalone server and replica set. For the standalone server, we use a 16GB NVDIMM-N server with 32 cores Intel Xeon E5-2640 2.60GHz processor, 32GB DRAM, and Samsung SSD 850 Pro as the storage device. For the replica set, the primary node is same as the standalone server, two secondary nodes have the same configuration with 4 cores processor, 4GB RAM and Samsung SSD 850 Pro as the storage device. All nodes have the same software layer with Linux kernel 4.4.0, MongoDB 3.2 with WiredTiger as storage engine and keep all settings as default. The experiment is setup by running 16 threads of YCSB 0.5.0 as the client request with 20 million of 1-KB documents with heavy update workflow (workload a with read:write ratio is 50:50) and the only update workflow to evaluate the effect of journaling configuration in MongoDB. We use various journaling settings as shown in Table 1.

<table>
<thead>
<tr>
<th>Methods</th>
<th>w</th>
<th>j</th>
<th>Journal option</th>
<th>Journal location</th>
</tr>
</thead>
<tbody>
<tr>
<td>w1j1</td>
<td>1</td>
<td>true</td>
<td>enable</td>
<td>same</td>
</tr>
<tr>
<td>w1j0</td>
<td>1</td>
<td>false</td>
<td>enable</td>
<td>same</td>
</tr>
<tr>
<td>w1j1NVM</td>
<td>1</td>
<td>true</td>
<td>enable</td>
<td>NVM</td>
</tr>
<tr>
<td>w1j0NVM</td>
<td>1</td>
<td>false</td>
<td>enable</td>
<td>NVM</td>
</tr>
<tr>
<td>w1nojournal</td>
<td>n/a</td>
<td></td>
<td>disable</td>
<td>same</td>
</tr>
</tbody>
</table>

W and j are the parameters of write concern setting in client request discussed in previous section; journal option is set in mongod.conf file to enable/disable journaling in MongoDB; we use journal location option to locate the journal files in the same location.
with data files on the SSD or on NVM devices i.e. NVDIMM.

3.2. Experiment results

We mainly analysis the performance on throughput with w1j1 as the base line method. Experiment results on the standalone server are illustrated in Figure 3. In general, write concern with journal set to false have higher throughput than journal set to true due to the server acknowledge to the request without waiting for log records persist in journal files. Locate journal files in NVDIMM improve the throughput up to 50% and 250% for heavy update workload and only update workload respectively. That imply that the higher intensive write from workload, the better throughput improvement. Because writes on NVDIMM is fast as on DRAM, there are two interested results: (1) w1j1NVM shows similar or even better performance compare to w1j0 with heavy update workload and only update workload respectively, and (2) w1j0NVM shows a little better performance compare to without journal method.

![Figure 3 Journaling evaluation on standalone server](image1)

![Figure 4 Journaling evaluation on the replica set](image2)

Figure 4 shows experiment results on the replica set. With the same data size, replica set shows higher throughput than standalone server up to 164% in case of w1j1NVM. Similar with standalone server, locate journal files in NVDIMM improve the performance up to 76% and 172% for heavy update workload and only update workload respectively. w1j0NVM also has throughput as high as with without journal is used i.e. w1nojournal.

4. Conclusion and Future Works

Our experiment shows that locating journal files in NVDIMM devices instead of same location in flash SSDs with data files dramatically improve the throughput in both standalone servers and replica sets. Only update workloads show better improvement than heavy update workloads. Those observations open an opportunity for the next research such that for high intensive write workload, MongoDB log records are transferred to low-latency high performance NVM devices to reduce the overhead of maintaining log records. We also consider experiment the proposal method in more complex data model with e.g. Linkbench. Further, we consider optimizing other components in MongoDB that have similar mechanism as log files e.g. metadata files, temporary files.

5. Acknowledgment

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the “SW Starlab” (IITP-2015-0-00314) supervised by the IITP (Institute for Information & communications Technology Promotion), and supported by Samsung Electronics in 2017.

REFERENCES