# A Step-by-step Tutorial to Reproduce *MonEx* Use Case Experiments

This file contains the experimental artifacts of the use case experiments of the *MonEx* paper. In the same folder of this file, you can find the experiments folders that contain the datasets and some dependent scripts.

## 1 Disk & Power Usage of a *MongoDB* Cluster

### 1.1 Objective

Using *MonEx* framework to monitor the disk & the power usage of a *MongoDB* cluster while running an indexation workload.

### 1.2 Requirements

The experiment has an automated script to run on the *Grid'5000* testbed, but we list the necessary steps to allow repeating it elsewhere.

#### 1.2.1 Hardware Requirements

This experiment needs a cluster of at least five nodes. Three nodes serve as shard nodes while another node holds the configuration server and the *mongo* router daemons. The last node is reserved for installing the *MonEx* framework. Additionally, the main point during reproducing this experiment is on the way we monitor the experiment using *MonEx* but not its results, so any hardware specifications could be used. Thus, the only impact is that you might obtain a different execution time.

#### 1.2.2 Software Requirements

Many software are necessary:

- *MonEx* configured with *Prometheus* as a data collector.
- *Prometheus* default node exporter
- *MongoDB* version 3.2
- *WiredTiger* search engine
- *Ruby (1.9 or above)*
- *mongo(v1.1), yaml* ruby-gems to generate data for *MongoDB*

### 1.3 Installation

For simplicity reasons, we refers to the experiment nodes as follows:

- node_[1-3]: the first, second & third shard nodes of *MongoDB*
- node_4: the node that holds the config. server and the router of *MongoDB*
- node_monex: the node that will be used to hold the *MonEx* framework.

#### 1.3.1 *MongoDB* Installation & Configuration

⇒ Please refer to the *MongoDB* official webpage to install *MongoDB* on the nodes_[1-4] and make sure to stop the *mongod* service on all nodes after the installation:
`https://docs.mongodb.com/v3.2/tutorial/install-mongodb-on-debian/`

⇒ Configure the *WiredTiger* to be used as the default storage engine in *MongoDB*. This is done by editing the file /etc/mongod.conf on the nodes_[1-4], by adding the following line:

```
storage.engine: wiredTiger
```

⇒ Configure the *MongoDB* cluster as the following:
- run the follwing commands on the node_4:

```
nohup mongod --configsvr --port 27019 &  // to start a config serv. daemon
nohup mongos --configdb [node_4_IP:27019] // to start a mongo router daemon
```

- run the following command on the nodes_[1-3]:

```
nuhup mongod --config /etc/mongod.conf --shardsvr --port 27017 &
```

⇒ We need to configure the cluster and to have a sharded data collection, so we run the following commands on the node_4 using *mongo* daemon:

```
$ mongos
mongos> sh.addShard([node_1:IP]) // repeat this step for node_2 & node_3
mongos> use db1   // create db1 database and jump inside
mongos> sh.enableSharding(db1)   // indicating that the db1 could be distributed
mongos> db.c1.ensureIndex({_id: hashed})  // create a free collection c1
mongos> sh.shardCollection(db1.c1, {_id:hashed})  // sharding the free collection c1
```

⇒ Create and inject a data into the configured cluster. To do so, we need to use the provided file (mongoXP/generateDumpsDataForMongoDB.rb) for generating data (20 million documents by default, about 80Gbyte). The script could be run on a standalone *MongoDB* and then we could create a dump of the generated data in order to be injected into our cluster using the *mongo* router. The commands on this steps are as follows:

```
ruby generateDumpsDataForMongoDB.rb  // on a machine running a mongod daemon
mongodump  --db db1 --collection c1   // export the collection data into a bson file
```

On the node_4, we execute the following command to inject the created data into our cluster

```
mongorestore -d db1 -c c1  [bson file]   // inject and evenly distribute the data
```

At the end of this step, *MongoDB* is installed and configured correctly.

### 1.3.2 *MonEx* installation

Clone the *MonEx* github project into the node_monex and install its dependencies as explained in the project page (`https://github.com/madynes/monex`). This step implies installing the default exporter of *Promtheus* into all the data nodes (nodes_[1-3]), and configuring *Prometheus* to regularly scrape those data nodes at the rate of 1 second.

## 1.4 Experiment workflow

We suppose that *MongoDB* is installed correctly and that *MonEx* server is started as explained in the previous steps. Then, we execute the following steps.

1. Before running the main workload of the experiment, notify *MonEx* that the indexation workload is to be started in a moment. Thus, the following command must be run on the node_4:

   ```
   curl -X POST MonEx_IP:5000/exp/mongoXP
   ```

2. We then run the workload from the same node as follows:

   ```
   $mongos
   mongos> use db1
   mongos> db.c1.createIndex({randNum:1})
   ```

   This will take several minutes regarding the hardware specification of the shard nodes, and the size of the generated data.

3. When the workload is accomplished, we should notify *MonEx* that the experiment is terminated. So, the following command is run dynamically from the experiment script (you can run it manually when you the previous command is terminated):

   ```
   curl -X PUT MonEx_IP:5000/exp/mongoXP
   ```

4. We can query the *MonEx* server to obtain a dataset containing the target metrics of the experiment. The commands to obtain the disk usage, and the power usage metrics are as follows:

```
curl -X GET -H "Content-Type: application/json" IP:5000/exp/mongoXP \
-d '{"query":"rate(node_disk_io_time_ms{device=\"sda1\", \
"server":"prometheus","type":"duration","labels":["instance"]}' > dataset_hdd.csv

curl -X GET -H "Content-Type: application/json" IP:5000/exp/mongoXP \
-d '{"query":"pdu","server":"prometheus","type":"duration",   \
"labels":["outlet","instance"]}' > dataset_pdu.csv
```

5. The obtained datasets from the previous step are available in the experiment folder. So if it is not possible to repeat the experiment, you can use the *MonEx*-draw over the obtained datasets to produce the figures of this experiment as follows:

```
./monex-draw -F2 dataset_pdu.csv -F dataset_hdd.csv
-c node1,node2,node3
-c2 node1,node2,node3
-x "Time (sec)" -y "Disk utilization (%)" -y2 "Power usage (W)" -l
```

This command produces the *Fig. 2-b* that is appeared in the *MonEx* paper.

# 2 Many-nodes Bittorrent Download

# 3 Input/output offset sequences experiment