MongoDB evaluation for calibration store

Content

Installation

Installation

```
on pslogin
ana-1.3.37
scs
cd ...
virtualenv venv-pymongo
source venv-pymongo/bin/activate

???
# python -m pip install pymongo


Alternative installation:
-------------------------
# https://docs.mongodb.com/manual/tutorial/install-mongodb-on-linux/
cd lib

curl -O https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-3.6.2.tgz

tar -zxvf mongodb-linux-x86_64-3.6.2.tgz

mkdir -p mongodb
cp -R -n mongodb-linux-x86_64-3.6.2/ mongodb

export PATH=/reg/neh/home/dubrovin/LCLS/venv-pymongo/lib/mongodb/mongodb-linux-x86_64-3.6.2/bin/:$PATH
echo $PATH
The same in
source set_path_to_mongodb

1. Create the data directory
mkdir -p ./data/db

2. Set r/w permissions for the data directory
chmod 775 data
chmod 775 data/db
```

Run server

Run server

```
pslogin
ssh psanaphi105
cd LCLS/venv-pymongo/
source bin/activate
source set_path_to_mongodb
assumes that ./data/db is already created
mongod --dbpath ./data/db --bind_ip_all &

!!! DO NOT CLOSE WINDOW,
```

Shell

Shell is a manual command line interface.

Shell

```
mongo --host psanaphi105 --port 27017

To exit the shell, type quit() or use the <Ctrl-C> shortcut.

> db
test

> show dbs
admin           0.000GB
calib-cxif5315  0.006GB
config          0.000GB
local           0.000GB

> use calib-cxif5315
switched to db calib-cxif5315

> show collections
cspad-0-cxids1-0
cspad-1

> db["cspad-0-cxids1-0"].find()
> db["cspad-0-cxids1-0"].find().pretty()

# Delete databale:
use calib-cxif5315
db.dropDatabase()

# Delete collection
db.collection.drop()
# OR:
db["cspad-0-cxids1-0"].drop()

> help

# Export/backup database in file
> mongodump -d <dbname> --archive <filename> --out /path/to/backup/dir

# Import database from file
mongorestore -d <dbname> --archive <filename>
```

Connection to DB in python

Connection to the data base

```
from pymongo import MongoClient
#client = MongoClient('localhost', 27017)
client = MongoClient('psanaphi105', 27017) #, username=uname, password=pwd)
db = client['calib-cxi12345']
col = db['camera-0-cxids1-0']
```

Connection time is 50-150ms depending on host and time.

Python API

client property and methods

```
address                 get_database            max_idle_time_ms        read_concern
arbiters                get_default_database    max_message_size        read_preference
close                   HOST                    max_pool_size           secondaries
close_cursor            is_locked               max_write_batch_size    server_info
codec_options           is_mongos               min_pool_size           server_selection_timeout
database_names          is_primary              next                    set_cursor_manager
drop_database           kill_cursors            nodes                   unlock
event_listeners         local_threshold_ms      PORT                    write_concern
fsync                   max_bson_size           primary
```

database property and methods

```
add_son_manipulator         error                         previous_error
add_user                    eval                          profiling_info
authenticate                get_collection                profiling_level
client                      incoming_copying_manipulators read_concern
codec_options               incoming_manipulators         read_preference
collection_names            last_status                   remove_user
command                     logout                        reset_error_history
create_collection           name                          set_profiling_level
current_op                  next                          system_js
dereference                 outgoing_copying_manipulators validate_collection
drop_collection             outgoing_manipulators         write_concern
```

collection property and methods

```
aggregate                   find_one                    next
bulk_write                  find_one_and_delete         options
codec_options               find_one_and_replace        parallel_scan
count                       find_one_and_update         read_concern
create_index                full_name                   read_preference
create_indexes              group                       reindex
database                    index_information           remove
delete_many                 initialize_ordered_bulk_op  rename
delete_one                  initialize_unordered_bulk_op replace_one
distinct                    inline_map_reduce           save
drop                        insert                      update
drop_index                  insert_many                 update_many
drop_indexes                insert_one                  update_one
ensure_index                list_indexes                with_options
find                        map_reduce                  write_concern
find_and_modify             name
```

documents = col.find(...) methods

```
add_option()        count()             max_time_ms()
address             cursor_id           min()
alive               distinct()          next()
batch_size()        explain()           remove_option()
clone()             hint()              retrieved
close()             limit()             rewind()
collation()         max()               skip()
collection          max_await_time_ms() sort()
comment()           max_scan()          where()
```

document properties

```
clear()      get()        pop()        update()
copy()       items()      popitem()    values()
fromkeys()   keys()       setdefault()
```

Tentative model of the calibration store

Experiment-centric calibration data base

Tentative schema of the experiment-centric db

```
# Database for experiment
dbexp = client("db-cxif5315")
fs = big_data_file_system(dbexp)

# Collections:
"cspad-0-cxids2-0"
"cspad2x2-0-cxids2-0"
"andor-0-cxids2-0"
...
# Auto-generated collections
'fs.files'
'fs.chunks'

# Document content for dbexp
doc = {
   "_id":ObjectId("53402597d852426020000002"), # auto-generated
   "experiment": "cxif5315",
   "run": "123",
   "detector": "cspad-0-cxids2-0",
   "ctype": "pedestals",
   "time_sec": "1516321053",
   "time_stamp": "2018-01-18T16:17:33-0800",
   "version": "v00",
   "uid": "login-name",
   "host": "psanaphi102",
   "port": "12345",
   "comment": "very good constants",
   "id_data": "5a98464a5777035bba3a4f41"  # added as a reference to big data
 }
```

All meta-data information is accessible through a single-level document.

Detector-centric calibration data base

Tentative schema for detector-centric db

```
# Model #1: DB per detector type, collection per detector:
-------------------------------------------------------
dbdet = client('db-cspad')

# Collections:
'cspad-0-cxids1-0'
'cspad-0-cxids2-0'
'cspad-0-cxidsd-0'
'cspad-0-xcsendstation-0'
'cspad-0-xppgon-0'
'cspad-0-sxrbeamline-1'
'cspad-0-mectargetchamber-0'

# Document content for dbdet the same as dbexp plus "id_data"
doc = {...
   "id_data": ObjectId("534009e4d852427820000002"),
   ...
}

# Model #2: DB per detector, one collection per detector:
-------------------------------------------------------
dbdet = client('db-cspad-0-cxids1-0')
col = dbdet['cspad-0-cxids1-0']

# Add collections in case of DB copy
'fs.files'
'fs.chunks'
```

Essentially document in the detector collection has a reference to the data in the experiment collections.

Data flow for documents less than 16 MB

Preparation of data

Conversion of numpy array to unicode

```
nda = gu.random_standard(shape=(32,185,388), mu=20, sigma=5, dtype=gu.np.float)

import pickle
from bson.binary import Binary

t0_sec = time()

arr = nda.flatten()
arr = ' '.join(['%.2f' % v for v in arr])
sarr = Binary(pickle.dumps(arr, protocol=2), subtype=128)

doc = {
   "experiment": "cxi12345",
   "run": 124,
   ...
   "data": sarr,
}

dt_sec = time() - t0_sec
```

- Preparation of cspad data in text/unicode format for inserting takes ~1sec.
- Only limited precision data can be saved due to limit on document size 16MB.

Inserting data

Insert document in collection

```
doc_id = col.insert_one(doc).inserted_id
```

Insertion time is 110-180ms.

Find data

Find data

```
t0_sec = time()
docs = col.find({"run": 125})
dt_sec = time() - t0_sec
```

Finding data time is 50-60us

Unpack data

Unpack data from unicode to numpy array

```
doc = docs[0]
xcarr = pickle.loads(doc["data"]) # 30-40ms
arr = gu.np.fromstring(xcarr, dtype=float, count=-1, sep=' ') # 300ms
```

Time to unpack is 350ms.

Data flow for large documents

Timing test is done for mongod running on psanaphi105 and scripts on psanagpu106.

Initialization

Initialization MongoDB and GridFS

```
import gridfs
from pymongo import MongoClient
#client = MongoClient('localhost')
client = MongoClient('psanaphi105', 27017)
db = client['calib-cxi12345']
fs = gridfs.GridFS(db)
col = db['camera-0-cxids1-0']
```

Time to connect 116-150ms.

Put

Save numpy array in db

```
ida = fs.put(nda.tobytes())
```

Time to save data 330-420ms.

Preparation of document with metadata and insert

```
doc = {
   "experiment": "cxi12345",
   "run": 126,
   "detector": col.name,
   "ctype": "pedestals",
   "data_size":  nda.size,
   "data_shape": nda.shape,
   "data_type":  str(nda.dtype),
   "data_id":    ida,
        ...
}
doc_id = col.insert_one(doc).inserted_id
```

Document meta-data with reference to data preparation time is 43-53us.

Insert metadata time 0.5-0.6ms.

Get

Find and get document

```
docs = col.find({"time_stamp" : "2018-01-25T09:33:10PST"})
doc = docs[0]
```

Metadata find and get time: 0.7ms

Get data from fs

```
s = fs.get(doc['data_id']).read()
nda = gu.np.fromstring(s)
```

Data extraction time: 96ms. Thus returned array is "flattend" and needs to be shaped.

Interface from Murali

2018-08-03 e-mail from Murali:
I have installed Mongo 4.0 on psdb-dev. I was hoping to use their REST service but this seems to have been deprecated and eliminated since 3.6.
So, I knocked a quick web service and have proxied it from pswww. This web service (https://github.com/slaclab/psdm_mongo_ws) is a suggestion only; please let me know if you need something different.
These are examples of getting data over HTTPS from a batch node from within cori; needless to say, the URL prefix is **https://pswww.slac.stanford.edu/calib_ws**
Two users:

- mongo --host=psdb-dev --port 9306 -u "dubrovin" -p "...." --authenticationDatabase "admin"
- mongo --host=psdb-dev --port 9306 -u "calibuser" -p "...." --authenticationDatabase "admin"

Test commands:

- curl -s "https://pswww.slac.stanford.edu/calib_ws/test_db/test_coll/5b649a9df59ae00bda110168"
- curl -s "https://pswww.slac.stanford.edu/calib_ws/test_db/test_coll"
- curl -s "https://pswww.slac.stanford.edu/calib_ws/test_db/test_coll?item=planner&size.uom=cm"
- curl -s "https://pswww.slac.stanford.edu/calib_ws/test_db/test_coll?query_string=%7B%20%22item%22%3A%20%22planner%22%2C%20%22qty%22%3A%2075%20%7D%0A"
- curl -s "https://pswww.slac.stanford.edu/calib_ws/" - get string of databases
- curl -s "https://pswww.slac.stanford.edu/calib_ws/test_db" - get list of collections in database
- curl -s "https://pswww.slac.stanford.edu/calib_ws/cdb_cxic0415/cspad_detnum1234?ctype=pedestals&data_size=2296960&run=74" - find and return document for query
- curl -s "https://pswww.slac.stanford.edu/calib_ws/cdb_cxic0415/cspad_detnum1234/gridfs/5b6893e81ead141643fe4344" - get document with constants from GridFS using document id
- curl -s "https://pswww.slac.stanford.edu/calib_ws/cdb_cxic0415/cspad_detnum1234/gridfs/5b6893e81ead141643fe4344" - DEPRICATED - access to GridFS raw data through doc _id
- curl -s "https://pswww.slac.stanford.edu/calib_ws/cdb_cxic0415/gridfs/5b6893d91ead141643fe3f6a" - access to GridFS raw data through data _id

Implementation

- Source code: https://github.com/slac-lcls/lcls2/tree/master/psana/psana/pscalib/calib see all modules `MDB*.py`
- Command Line Interface (CLI), command `cdb`: https://github.com/slac-lcls/lcls2/blob/master/psana/psana/pscalib/app/cdb.py
- Graphical User Interface (GUI), command `calibman`: https://github.com/slac-lcls/lcls2/blob/master/psana/psana/graphqt/app/calibman.py
- Web-service access interface
  - Python: https://github.com/slac-lcls/lcls2/blob/master/psana/psana/pscalib/calib/MDBWebUtils.py
  - C++: https://github.com/slac-lcls/lcls2/blob/master/psalg/psalg/calib/MDBWebUtils.hh

Write web access

2019-07-27 web service to write in DB

```
2019-07-27
Here's version 1; any feedback is appreciated.
Regards,
Murali

#!/usr/bin/env python

"""
Sample for posting to the calibration service using a web service and kerberos authentication.
Make sure we have a kerberos ticket.
"""

import requests
import json
from krtc import KerberosTicket
from urllib.parse import urlparse

ws_url = "https://pswww.slac.stanford.edu/ws-kerb/calib_ws/"
krbheaders = KerberosTicket("HTTP@" + urlparse(ws_url).hostname).getAuthHeaders()

# Create a new document in the collection test_coll in the database test_db.
resp = requests.post(ws_url+"test_db/test_coll/", headers=krbheaders, json={"calib_count": 1})
print(resp.text)
new_id = resp.json()["_id"]

# Update an existing document
resp = requests.put(ws_url+"test_db/test_coll/"+new_id, headers=krbheaders, json={"calib_count": 2})
print(resp.text)

# Delete an existing document
resp = requests.delete(ws_url+"test_db/test_coll/"+new_id, headers=krbheaders)
print(resp.text)

# Create a new GridFS document, we upload an image called small_img.png
files = [("files",  ('small_img.png', open('small_img.png', 'rb'), 'image/png'))]
resp = requests.post(ws_url+"test_db/gridfs/", headers=krbheaders, files=files)
print(resp.text)
new_id = resp.json()["_id"]

# Delete the GridFS document
resp = requests.delete(ws_url+"test_db/gridfs/"+new_id, headers=krbheaders)
print(resp.text)
```

Summary

- MongoDB structure has limitations in number of levels and document size.
  - server may have many DBs
  - DB is a container for collections
  - collection is a group of documents

    - document is a JSON/BSON object of key:value pairs (dictionary). Each value may be dictionary itself etc, but further structure levels are not supported by DB structure.
        - document size has hardwired limit 16MB (in 2010 increased from 4 to 16MB and devs do not want to change it). CSPAD 2Mpix*8byte(double) = 16MB, but we may expect larger detectors like Jungfrau, Epix, Andor, etc.
        - Larger data size is suggested to save using GridFS; split data for chanks and save chunks in the same DB in different collections.
        - JSON (text) object in MongoDB is presented in unicode...(UTF-8). Data should be converted to unicode force and back in saving retrieving.
- schema-less DB looks interesting to certain extents, but in order to find something in DB there should be a schema...
- GridFS works fine with document size>16GB.

References

- install-mongodb-on-linux
- tutorial
- recover-data-following-unexpected-shutdown
- authorization