openleveldb

Release 0.1.0

Luca Moschella

Apr 15, 2021

CONTENTS

Openleveldb is a small pythonic wrapper around Plyvel

CHAPTER

ONE

FEATURES

1.1 Transparent object store

It works with python objects:

- Automatically **encodes objects** into bytes when saving to leveldb
- Automatically **decodes bytes** into their original type when retrieving objects from leveldb

Supported types include:

- `int`
- `str`
- `numpy.ndarray`
- Anything that is serializable by orjson

```
>>> db['key'] = {'key': [1, 2, 3]}
>>> db['key']
{'key': [1, 2, 3]}
```

1.2 Python dict-like protocol

It offers dict-like interface to LevelDB

```
>>> db["prefix", "key"] = np.array([1, 2, 3], dtype=np.int8)
>>> db["prefix", "key"]
array([1, 2, 3], dtype=int8)
```

```
>>> db = db["prefix", ...]
>>> db["key"]
array([1, 2, 3], dtype=int8)
```

1.3 String-only keys

The only possible type for the keys is `str`. It avoids several problems when working with prefixes.

1.4 Multiprocessing support

Experimental **multiprocessing** support using a background flask server, exposing the same API of a direct connection:

```
db = LevelDB(db_path="path_to_db", server_address="http://127.0.0.1:5000")
```

1.4.1 Installation

It is possible to install `openleveldb` with poetry:

```
poetry add openleveldb
```

or with pip:

```
pip install openleveldb
```

Verify installation

Verify that the installation has been successful and that `plyvel` correctly installed leveldb, if it is not already installed on the system:

```
python -c 'import openleveldb'
```

Verify that `openleveldb` using the tests

```
git clone git@github.com:lucmos/openleveldb.git
cd openleveldb
poetry run pytest .
```

1.4.2 Getting started

Open db connection

The connection to the db can be direct or pass through a REST server. The only change required in the code is how the `LevelDB` object is instantiated

Direct connection

The first thing to do is to instantiate a `LevelDB` object to open a connection to leveldb database:

```
from openleveldb import LevelDB
db = LevelDB(db_path="path_to_db")
```

REST connection

If it's required to have multiprocessing support, that is not provided by leveldb, it is possible to start a server and connect to the database through REST API. In order to start the server is enough to do:

```
cd openleveldb
make server
```

Then it's possible to instantiate a `LevelDB` object specifying the server:

```
from openleveldb import LevelDB
db = LevelDB(db_path="path_to_db", server_address="http://127.0.0.1:5000")
```

Basic access

Storing, reading and deleting an element follow the dict protocol:

```
>>> db["prefix", "key"] = np.array([1, 2, 3], dtype=np.int8)
>>> db["prefix", "key"]
array([1, 2, 3], dtype=int8)
>>> del db["prefix", "key"]
```

It is possible to use an arbitrary number of prefixes:

```
>>> db["prefix1", "prefix2", "key"] = np.array([1, 2, 3], dtype=np.int8)
>>> db["prefix1", "prefix2", "key"]
array([1, 2, 3], dtype=int8)
>>> del db["prefix1", "prefix2", "key"]
```

Iteration

Iteration over `(key, value)` pairs behaves accordingly:

```
>>> list(db)
[('a1', 'value1'), ('b1', 'value2'), ('b2', 'value3'), ('c1', 'value4')]
```

It's possible to perform advanced form of iteration using the `LevelDB.prefixed_iter` function:

```
>>> list(db)
[('a1', 'value1'), ('b1', 'value2'), ('b2', 'value3'), ('c1', 'value4')]
>>> list(db.prefixed_iter(prefixes=["b"]))
[('1', 'value2'), ('2', 'value3')]
>>> list(db.prefixed_iter(prefixes=["b", "1"]))
[('', 'value2')]
>>> list(db.prefixed_iter(starting_by="b"))
```

(continues on next page)

(continued from previous page)

```
[('b1', 'value2'), ('b2', 'value3')]
>>> list(db.prefixed_iter(starting_by=["b", "1"]))
[('b1', 'value2')]
```

Fancy indexing

When a local connection is available, it is possible to use fancy indexing to obtain a stateful `LevelDB` that remembers the prefixes:

```
>>> list(db)
[('a1', 'value1'), ('b1', 'value2'), ('b2', 'value3'), ('c1', 'value4')]
>>> db_b = db['b', ...]
>>> db_b["1"]
'value2'
>>> list(db_b)
[('1', 'value2'), ('2', 'value3')]
>>> list(db["c", ...])
[('1', 'value4')]
```

1.4.3 Public API