Sharkbite Documentation

Release 0.7.2

Marc Parisi

Feb 24, 2021

Contents

1 About Sharkbite **3**
- 1.1 About the name . . . 3
- 1.2 Installing . . . 3
- 1.3 Features . . . 4
- 1.4 HDFS Client . . . 4
- 1.5 Version Detection . . . 4
- 1.6 Hedged Reads . . . 4
- 1.7 Python Iterators . . . 4

2 Async Example **7**

3 HDFS Client **11**

4 Getting Statistics in Python **13**

5 Sharkbite Documentation **15**

6 Indices and tables **17**

CHAPTER 1

About Sharkbite

Sharkbite is an HDFS and native client for key/value stores. With initial support for Apache Accumulo Accumulo, the design can and has been used to support other key/value stores. Development began in 2014 and has slowly evolved. there is no structural specificity to Accumulo despite being the base implementation. Despite this the examples below will look very much like Accumulo due to aliasing. This is intentional.

Capabilities That will be supported in V1.0 :

- Works with Accumulo 1.6.x, 1.7.x, 1.8.x, 1.9.x and 2.x
- **Read/Write** : Reading and writing data to Accumulo is currently supported.
- **Table Operations** : Most table operations are currently supported. This includes the fate operations that the normal Accumulo client performs.
- **Security Operations** : Most security operations are implemented. Please let us know of any gaps.

1.1 About the name

Sharkbite's name originated from design as a connector that abstracted components in which we tightly coupled and gripped interfaces of the underlying datastore. With an abstraction layer for access, and using cross compatible objects, the underlying interfaces are heavily coupled to each database. As a result, Sharkbite became a fitting name since interfaces exist to abstract the high coupling that exists within implementations of the API.

1.2 Installing

This python client can be installed via *pip install sharkbite*

A Python example is included. This is your primary example of the Python bound sharkbite library.

Sharkbite supports async iteration A simple example is provided.

1.3 Features

1.4 HDFS Client

Sharkbite supports a limited HDFS client. As this functionality grows so will the capabilities. Version 0.7 will support a complete HDFS client. Since Sharkbite it built as a python bindings around a C++ Client, the python client will mature slightly behind the C++ client, hence the delta with building this into V 0.7

1.5 Version Detection

Sharkbite detects the version of Apache Accumulo. Therefore you will be able to simply create a connector to the zookeeper instance.

1.6 Hedged Reads

Sharkbite supports hedged reads (executing scans against RFiles when they can be accessed) concurrently with Accumulo RPC scans. The first executor to complete will return your results. This feature is in beta and not suggested for production environments.

Enable it with the following option:

```
    import pysharkbite as sharkbite

    connector = sharkbite.AccumuloConnector(user, zk)

table_operations = connector.tableOps(table)

    scanner = table_operations.createScanner(auths, 2)

range = sharkbite.Range("myrow")

scanner.addRange( range )

### enable the beta option of hedged reads

scanner.setOption( sharkbite.ScannerOptions.HedgedReads )

resultset = scanner.getResultSet()

for keyvalue in resultset:
    key = keyvalue.getKey()
    value = keyvalue.getValue()
```

1.7 Python Iterators

We now support a beta version of python iterators. By using the cmake option PYTHON_ITERATOR_SUPPORT (cmake -DPYTHON_ITERATOR_SUPPORT=ON) we will build the necessary infrastructure to support python iterators

Iterators can be defined as single function lambdas or by implementing the seek or next methods.

The first example implements the seek and onNext methods. seek is optional if you don't wish to adjust the range. Once keys are being iterated you may get the top key. You may call iterator.next() after or the infrastructure will do that for you.

```
class myIterator:
  def seek(iterator,soughtRange):
    range = Range("a")
    iterator.seek(range)


  def onNext(iterator):
    if (iterator.hasTop()):
        kv = KeyValue()
          key = iterator.getTopKey()
          cf = key.getColumnFamily()
          value = iterator.getTopValue()
          key.setColumnFamily("oh changed " + cf)
          iterator.next()
          return KeyValue(key,value)
    else:
      return None
```

If this is defined in a separate file, you may use it with the following code snippet

```
with open('test.iter', 'r') as file:
 iterator = file.read()
## name, iterator text, priority
 iterator = pysharkbite.PythonIterator("PythonIterator",iteratortext,100)
 scanner.addIterator(iterator)
```

Alternative you may use lambdas. The lambda you provide will be passed the KeyValue (getKey() and getValue() return the constituent parts). A partial code example of setting it up is below. You may return a Key or KeyValue object. If you return the former an empty value will be return ed.

```
## define only the name and priority
iterator = pysharkbite.PythonIterator("PythonIterator",100)
## define a lambda to ajust the column family.
iterator = iterator.onNext("lambda x : Key( x.getKey().getRow(), 'new cf', x.getKey().
↪getColumnQualifier()) ")

scanner.addIterator(iterator)
```

You may either define a python iterator as a text implementation or a lambda. Both cannot be used simulaneously.

[accumulo]: https://accumulo.apache.org

CHAPTER 2

Async Example

Below is an example client based on pysharkbite. We'll step through important aspects followed by the entirety of the code at the end.

The first interesting piece of code that we come across is printasync, which is intended to asynchronously print the rows from all keys. This coroutine will be used later to loop through the asynchronous iterator.

```
async def printasync(iter):
    async for keyvalue in iter:
        key = keyvalue.getKey()
        print(key.getRow())
```

After writing data the example creates a scanner. This scanner creates a range from 'row' to 'row3' then creates an async event loop to call the coroutine printasync. As stated above this enables us to asynchronously print the rows.

```
scanner = table_operations.createScanner(auths, 2)

range = pysharkbite.Range("row"",True,"row3"",False)

scanner.addRange( range )

resultset = scanner.getResultSet()

loop = asyncio.get_event_loop()
loop.run_until_complete(printasync(resultset))
```

```
#!/usr/bin/python
# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements.  See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership.  The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License.  You may obtain a copy of the License at
#
```

(continues on next page)

(continued from previous page)

```
#   http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing,
# software distributed under the License is distributed on an
# "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
# KIND, either express or implied.  See the License for the
# specific language governing permissions and limitations
# under the License.
from ctypes import cdll
from argparse import ArgumentParser
from ctypes import cdll
import ctypes
import traceback
import time
import asyncio


"""

This is an Example of using the Python connectors. The example will accept user input
create a table writing arbitrary information to it via the BatchWriter and scanner␣
↪will put the written data


"""

parser = ArgumentParser(description="This is an Apache Accummulo Python connector")

parser.add_argument("-i", "--instance", dest="instance",
                    help="Apache Accumulo Instance Name", required=True)
parser.add_argument("-z", "--zookeepers", dest="zookeepers",
                    help="Comma Separated Zookeeper List", required=True)
parser.add_argument("-u", "--username", dest="username",
                    help="User to access Apache Accumulo", required=True)
parser.add_argument("-p", "--password", dest="password",
                    help="Password to access Apache Accumulo. May also be supplied at␣
↪the command line")
parser.add_argument("-t", "--table", dest="table",
                    help="Table to create/update")
args = parser.parse_args()

password = args.password
table = args.table

async def printasync(iter):
    async for keyvalue in iter:
        key = keyvalue.getKey()
        print(key.getRow())

if not password:
    print("Please enter your password")
    password = input()

if not table:
    table = "blahblahd"

import pysharkbite
```

(continues on next page)

(continued from previous page)

```
configuration = pysharkbite.Configuration()

zk = pysharkbite.ZookeeperInstance(args.instance, args.zookeepers, 1000,␣
↪configuration)

user = pysharkbite.AuthInfo(args.username, password, zk.getInstanceId())

try:
    connector = pysharkbite.AccumuloConnector(user, zk)


    table_operations = connector.tableOps(table)

    if not table_operations.exists(False):
        print ("Creating table " + table)
        table_operations.create(False)
    else:
        print (table + " already exists, so not creating it")


    auths = pysharkbite.Authorizations()

    """ Add authorizations """
    """ mutation.put("cf","cq","cv",1569786960) """

    writer = table_operations.createWriter(auths, 10)

    mutation = pysharkbite.Mutation("row2");

    mutation.put("cf","cq","",1569786960, "value")
    mutation.put("cf2","cq2","",1569786960, "value2")
    """ no value """
    mutation.put("cf3","cq3","",1569786960, "")

    writer.addMutation( mutation )

    writer.close()

    time.sleep(2)

    """ auths.addAuthorization("cv") """

    scanner = table_operations.createScanner(auths, 2)

    range = pysharkbite.Range("row"",True,"row3"",False)

    scanner.addRange( range )

    resultset = scanner.getResultSet()

    loop = asyncio.get_event_loop()
    loop.run_until_complete(printasync(resultset))


    """ delete your table if user did not create temp """
    if not args.table:
```

(continues on next page)

(continued from previous page)

```
        table_operations.remove()

except RuntimeError as e:
    traceback.print_exc()
    print("Oops, error caused: " + str(e))
```

CHAPTER 3

HDFS Client

The hdfs client is nearly full client. It lacks features found in the C++ client. These will be added over time. Please visit the link, above, to find the API of what is currently supported

An example usage of these functions is below. Note that if Opening RFiles in pysharkbite, you must specify the full path including the hdfs protocol if it is located on HDFS. This will open a full HDFS client to access these files.

```python
import pysharkbite

hdfs = pysharkbite.Hdfs("hdfs://namenode:8020",8020);

hdfs.mkdir("/directoryA/directoryB");

hdfs.list("/");
```

CHAPTER 4

Getting Statistics in Python

You can retreive Accumulo stats with the get_statistics function in the connector.

```
from ctypes import cdll
from argparse import ArgumentParser
from ctypes import cdll
import ctypes
import traceback
import json
import time



"""

This is an Example of using the Python connectors. The example will accept user input
create a table writing arbitrary information to it via the BatchWriter and scanner␣
↪will put the written data


"""

parser = ArgumentParser(description="This is an Apache Accummulo Python connector")

parser.add_argument("-i", "--instance", dest="instance",
                    help="Apache Accumulo Instance Name", required=True)
parser.add_argument("-z", "--zookeepers", dest="zookeepers",
                    help="Comma Separated Zookeeper List", required=True)
parser.add_argument("-u", "--username", dest="username",
                    help="User to access Apache Accumulo", required=True)
parser.add_argument("-p", "--password", dest="password",
                    help="Password to access Apache Accumulo. May also be supplied at␣
↪the command line")
parser.add_argument("-t", "--table", dest="table",
                    help="Table to create/update")
```

(continues on next page)

(continued from previous page)

```
args = parser.parse_args()

password = args.password
table = args.table

if not password:
    print("Please enter your password")
    password = input()

if not table:
    table = "blahblahd"

import pysharkbite

configuration = pysharkbite.Configuration()

zk = pysharkbite.ZookeeperInstance(args.instance, args.zookeepers, 1000,␣
↪configuration)

user = pysharkbite.AuthInfo(args.username, password, zk.getInstanceId())

try:
    connector = pysharkbite.AccumuloConnector(user, zk)

    stats = connector.getStatistics()

    print("Goal state: " + str(stats.goal_state))

    # print some server info
    for serverinfo in stats.tablet_server_info:
            print(serverinfo.name)
            print(serverinfo.last_contact)



except RuntimeError as e:
    traceback.print_exc()
    print("Oops, error caused: " + str(e))
```

CHAPTER 5

Sharkbite Documentation

CHAPTER 6

Indices and tables

- genindex
- modindex
- search