zict Documentation

Release 0.1.4

Matthew Rocklin

Apr 09, 2019

Contents

The dictionary / mutable mapping interface is powerful and multi-faceted.

- We store data in different locations such as in-memory, on disk, in archive files, etc..
- We manage old data with different policies like LRU, random eviction, etc..
- We might encode or transform data as it arrives or departs the dictionary through compression, encoding, etc..

To this end we build abstract `MutableMapping` classes that consume and build on other `MutableMappings`. We can compose several of these with each other to form intuitive interfaces over complex storage systems policies.

CHAPTER 1

Example

In the following example we create an LRU dictionary backed by pickle-encoded, zlib-compressed, directory of files.

```
import pickle
import zlib

from zict import File, Func, LRU

a = File('myfile/', mode='a')
b = Func(zlib.compress, zlib.decompress, a)
c = Func(pickle.dumps, pickle.loads, b)
d = LRU(100, c)

>>> d['x'] = [1, 2, 3]
>>> d['x']
[1, 2, 3]
```

CHAPTER 2

API

class zict.buffer.**Buffer**(*fast*, *slow*, *n*, *weight=<function <lambda>>*, *fast_to_slow_callbacks=None*, *slow_to_fast_callbacks=None*)

Buffer one dictionary on top of another

This creates a MutableMapping by combining two MutableMappings, one that feeds into the other when it overflows, based on an LRU mechanism. When the first evicts elements these get placed into the second. When an item is retrieved from the second it is placed back into the first.

Parameters

fast: MutableMapping

slow: MutableMapping

fast_to_slow_callbacks: list of callables These functions run every time data moves from the fast to the slow mapping. They take two arguments, a key and a value

slow_to_fast_callbacks: list of callables These functions run every time data moves form the slow to the fast mapping.

See also:

LRU

Examples

```
>>> fast = dict()
>>> slow = Func(dumps, loads, File('storage/'))  # doctest: +SKIP
>>> def weight(k, v):
...     return sys.getsizeof(v)
>>> buff = Buffer(fast, slow, 1e8, weight=weight)  # doctest: +SKIP
```

close()

Release any system resources held by this object.

items() → list of D's (key, value) pairs, as 2-tuples

keys() → list of D's keys

values() → list of D's values

class zict.file.**File**(*directory*, *mode='a'*)

Mutable Mapping interface to a directory

Keys must be strings, values must be bytes

Note this shouldn't be used for interprocess persistence, as keys are cached in memory.

Parameters

directory: string

mode: string, ('r', 'w', 'a'), defaults to 'a'

Examples

```
>>> z = File('myfile')  # doctest: +SKIP
>>> z['x'] = b'123'  # doctest: +SKIP
>>> z['x']  # doctest: +SKIP
b'123'
```

Also supports writing lists of bytes objects

```
>>> z['y'] = [b'123', b'4567']  # doctest: +SKIP
>>> z['y']  # doctest: +SKIP
b'1234567'
```

Or anything that can be used with file.write, like a memoryview

```
>>> z['data'] = np.ones(5).data  # doctest: +SKIP
```

keys() → list of D's keys

class zict.func.**Func**(*dump*, *load*, *d*)

Buffer a MutableMapping with a pair of input/output functions

Parameters

dump: callable Function to call on value as we set it into the mapping

load: callable Function to call on value as we pull it from the mapping

d: MutableMapping

Examples

```
>>> def double(x):
...     return x * 2
```

```
>>> def halve(x):
...     return x / 2
```

```
>>> d = dict()
>>> f = Func(double, halve, d)
>>> f['x'] = 10
```

(continues on next page)

(continued from previous page)

```
>>> d
{'x': 20}
>>> f['x']
10.0
```

close()
Release any system resources held by this object.

items() → list of D's (key, value) pairs, as 2-tuples

keys() → list of D's keys

values() → list of D's values

class zict.lmdb.**LMDB**(*directory*)
Mutable Mapping interface to a LMDB database.

Keys must be strings, values must be bytes

Parameters

directory: string

Examples

```
>>> z = LMDB('/tmp/somedir/')  # doctest: +SKIP
>>> z['x'] = b'123'  # doctest: +SKIP
>>> z['x']  # doctest: +SKIP
b'123'
```

close()
Release any system resources held by this object.

items() → list of D's (key, value) pairs, as 2-tuples

keys() → list of D's keys

values() → list of D's values

class zict.lru.**LRU**(*n*, *d*, *on_evict=None*, *weight=<function <lambda>>*)
Evict Least Recently Used Elements

Parameters

n: int Number of elements to keep, or total weight if weight= is used

d: MutableMapping Dictionary in which to hold elements

on_evict: list of callables Function:: k, v -> action to call on key value pairs prior to eviction

weight: callable Function:: k, v -> number to determine the size of keeping the item in the mapping. Defaults to `(k, v) -> 1`

Examples

```
>>> lru = LRU(2, dict(), on_evict=lambda k, v: print("Lost", k, v))
>>> lru['x'] = 1
>>> lru['y'] = 2
```

(continues on next page)

(continued from previous page)

```
>>> lru['z'] = 3
Lost x 1
```

close()
Release any system resources held by this object.

evict()
Evict least recently used key

This is typically called from internal use, but can be externally triggered as well.

Returns

k: key

v: value

w: weight

items() → list of D's (key, value) pairs, as 2-tuples

keys() → list of D's keys

values() → list of D's values

class zict.sieve.**Sieve**(*mappings*, *selector*)
Store values in different mappings based on a selector's output.

This creates a MutableMapping combining several underlying MutableMappings for storage. Items are dispatched based on a selector function provided by the user.

Parameters

mappings: dict of {mapping key: MutableMapping}

selector: callable (key, value) -> mapping key

See also:

Buffer

Examples

```
>>> small = {}
>>> large = DataBase()                          # doctest: +SKIP
>>> mappings = {True: small, False: large}      # doctest: +SKIP
>>> def is_small(key, value):                   # doctest: +SKIP
        return sys.getsizeof(value) < 10000
>>> d = Sieve(mappings, is_small)               # doctest: +SKIP
```

close()
Release any system resources held by this object.

items() → list of D's (key, value) pairs, as 2-tuples

keys() → list of D's keys

values() → list of D's values

class zict.zip.**Zip**(*filename*, *mode='a'*)
Mutable Mapping interface to a Zip file

Keys must be strings, values must be bytes

Parameters

filename: string

mode: string, ('r', 'w', 'a'), defaults to 'a'

Examples

```
>>> z = Zip('myfile.zip')  # doctest: +SKIP
>>> z['x'] = b'123'  # doctest: +SKIP
>>> z['x']  # doctest: +SKIP
b'123'
>>> z.flush()  # flush and write metadata to disk  # doctest: +SKIP
```

items() → list of D's (key, value) pairs, as 2-tuples

keys() → list of D's keys

values() → list of D's values

Index