python-lz4 Documentation

Release 0.10.0

Jonathan Underwood

Jun 15, 2017

Contents

CHAPTER 1

Contents

Introduction

This package provides a Python interface for the LZ4 compression library by Yann Collet. Support is provided for Python 2 (from 2.6 onwards) and Python 3 (from 3.3 onwards).

The LZ4 library provides support for three specifications:

- The frame format
- The block format
- The stream format

This Python interface currently supports the block format. Support for the frame format is also included as an unstable technology preview. Support for the streaming format will be available in a future release.

Install

The package is hosted on PyPI:

```
$ pip install lz4
$ easy_install lz4
```

Quickstart

The recommended binding to use is the LZ4 frame format. The simplest way to use the frame package is to import the compress and decompress functions:

```
>>> import os
>>> from lz4.frame import compress, decompress
>>> input_data = os.urandom(20 * 128 * 1024)  # Read 20 * 128kb
>>> compressed = compress(input_data)
>>> decompressed = decompress(compressed)
>>> decompressed == input_data
Out[6]: True
```

User Guide

lz4 package

Most of the functionality of this package is found in the *lz4.frame sub-package* and the *lz4.block sub-package* format bindings. The base package functionality is described here.

Contents

lz4.**lz4version**()
Returns the version number of the LZ4 library

Deprecated functions

The following methods are provided as wrappers around `compress` and `decompress` for backwards compatibility, but will be removed in the near future. These methods are also imported into the lz4 top level namespace for backwards compatibility.

This documentation is intentionally vague so as to discourage you from using these functions. Code that uses the functions will raised a `DeprecatedWarning`.

- `dumps`, and `LZ4_compress` are wrappers around `lz4.block.compress`
- `loads`, `uncompress`, `LZ4_uncompress` are wrappers around `lz4.block.decompress`
- `compress_fast` and `LZ4_compress_fast` are wrappers around `lz4.block.compress` with `mode=fast`
- `compressHC` is a wrapper around `lz4.block.compress` with `mode=high_compression`

lz4.frame sub-package

This sub-package is an incomplete technology preview.

This sub-package provides the capability to compress and decompress data using the LZ4 frame specification.

The frame specification is recommended for most applications. A key benefit of using the frame specification (compared to the block specification) is interoperability with other implementations.

Contents

A Python wrapper for the LZ4 frame protocol

lz4.frame.**compress**(*source*, *compression_level=0*, *block_size=0*, *content_checksum=0*, *block_mode=0*, *frame_type=0*, *content_size_header=1*)
Accepts a string, and compresses the string in one go, returning the compressed string as a string of bytes. The compressed string includes a header and endmark and so is suitable for writing to a file.

Parameters **source** (*str*) – String to compress

Keyword Arguments

- **block_size** (*int*) – Sepcifies the maximum blocksize to use. Options:
  - lz4.frame.BLOCKSIZE_DEFAULT or 0: the lz4 library default

  - lz4.frame.BLOCKSIZE_MAX64KB or 4: 64 kB
  - lz4.frame.BLOCKSIZE_MAX256KB or 5: 256 kB
  - lz4.frame.BLOCKSIZE_MAX1MB or 6: 1 MB
  - lz4.frame.BLOCKSIZE_MAX4MB or 7: 4 MB

  If unspecified, will default to lz4.frame.BLOCKSIZE_DEFAULT.

- **block_mode** (*int*) – Specifies whether to use block-linked compression. Options:
  - lz4.frame.BLOCKMODE_LINKED or 0: linked mode
  - lz4.frame.BLOCKMODE_INDEPENDENT or 1: disable linked mode

  The default is lz4.frame.BLOCKMODE_LINKED.

- **compression_level** (*int*) – Specifies the level of compression used. Values between 0-16 are valid, with 0 (default) being the lowest compression (0-2 are the same value), and 16 the highest. Values above 16 will be treated as 16. Values between 4-9 are recommended. The following module constants are provided as a convenience:
  - lz4.frame.COMPRESSIONLEVEL_MIN: Minimum compression (0, the default)
  - lz4.frame.COMPRESSIONLEVEL_MINHC: Minimum high-compression mode (3)
  - lz4.frame.COMPRESSIONLEVEL_MAX: Maximum compression (16)
- **content_checksum** (*int*) – Specifies whether to enable checksumming of the payload content. Options:
  - lz4.frame.CONTENTCHECKSUM_DISABLED or 0: disables checksumming
  - lz4.frame.CONTENTCHECKSUM_ENABLED or 1: enables checksumming

  The default is CONTENTCHECKSUM_DISABLED.

- **frame_type** (*int*) – Specifies whether user data can be injected between frames. Options:
  - lz4.frame.FRAMETYPE_FRAME or 0: disables user data injection
  - lz4.frame.FRAMETYPE_SKIPPABLEFRAME or 1: enables user data injection

  The default is lz4.frame.FRAMETYPE_FRAME.

- **content_size_header** (*bool*) – Specifies whether to include an optional 8-byte header field that is the uncompressed size of data included within the frame. Including the content-size header is optional and is enabled by default.

Returns Compressed data as a string

Return type str

lz4.frame.**decompress**(*source*)
Decompressed a frame of data and returns it as a string of bytes. :param source: LZ4 frame as a string :type source: str

Returns Uncompressed data as a string.

Return type str

lz4.frame.**create_compression_context**()
Creates a Compression Context object, which will be used in all compression operations.

Returns A compression context

Return type cCtx

lz4.frame.**compress_begin**()

compress_begin(cCtx, source_size=0, compression_level=0, block_size=0, content_checksum=0, content_size=1, block_mode=0, frame_type=0, auto_flush=1)

Creates a frame header from a compression context.

Parameters **context** (*cCtx*) – A compression context.

Keyword Arguments

- **block_size** (*int*) – Sepcifies the maximum blocksize to use. Options:
  - lz4.frame.BLOCKSIZE_DEFAULT or 0: the lz4 library default
  - lz4.frame.BLOCKSIZE_MAX64KB or 4: 64 kB
  - lz4.frame.BLOCKSIZE_MAX256KB or 5: 256 kB
  - lz4.frame.BLOCKSIZE_MAX1MB or 6: 1 MB
  - lz4.frame.BLOCKSIZE_MAX4MB or 7: 4 MB

  If unspecified, will default to lz4.frame.BLOCKSIZE_DEFAULT.
- **block_mode** (*int*) – Specifies whether to use block-linked compression. Options:
  - lz4.frame.BLOCKMODE_LINKED or 0: linked mode
  - lz4.frame.BLOCKMODE_INDEPENDENT or 1: disable linked mode

  The default is lz4.frame.BLOCKMODE_LINKED.
- **compression_level** (*int*) – Specifies the level of compression used. Values between 0-16 are valid, with 0 (default) being the lowest compression (0-2 are the same value), and 16 the highest. Values above 16 will be treated as 16. Values between 4-9 are recommended. The following module constants are provided as a convenience:
  - lz4.frame.COMPRESSIONLEVEL_MIN: Minimum compression (0, the default)
  - lz4.frame.COMPRESSIONLEVEL_MINHC: Minimum high-compression mode (3)
  - lz4.frame.COMPRESSIONLEVEL_MAX: Maximum compression (16)
- **content_checksum** (*int*) – Specifies whether to enable checksumming of the payload content. Options:
  - lz4.frame.CONTENTCHECKSUM_DISABLED or 0: disables checksumming
  - lz4.frame.CONTENTCHECKSUM_ENABLED or 1: enables checksumming

  The default is CONTENTCHECKSUM_DISABLED.
- **frame_type** (*int*) – Specifies whether user data can be injected between frames. Options:
  - lz4.frame.FRAMETYPE_FRAME or 0: disables user data injection
  - lz4.frame.FRAMETYPE_SKIPPABLEFRAME or 1: enables user data injection

  The default is lz4.frame.FRAMETYPE_FRAME.
- **auto_flush** (*int*) – Enable (1, default) or disable (0) autoFlush. When autoFlush is disabled, the LZ4 library may buffer data until a block is full
- **source_size** (*int*) – This optionally specifies the uncompressed size of the source content. This arument is optional, but if specified will be stored in the frame header for use during decompression.

Returns **str** – Frame header.

Return type str

lz4.frame.**compress_update**(*context*, *source*)
Compresses blocks of data and returns the compressed data in a string of bytes. :param context: compression context :type context: cCtx :param source: data to compress :type source: str

Returns Compressed data as a string

Return type str

Notes

If autoFlush is disabled (auto_flush=0 when calling compress_begin) this function may return an empty string if LZ4 decides to buffer. the input.

lz4.frame.**compress_end**(*context*)
Flushes a compression context returning an endmark and optional checksum as a string of bytes. :param context: compression context :type context: cCtx

Returns Remaining (buffered) compressed data, end mark and optional checksum as a string

Return type str

lz4.frame.**get_frame_info**(*frame*)
Given a frame of compressed data, returns information about the frame. :param frame: LZ4 frame as a string :type frame: str

Returns

Dictionary with keys blockSizeID, blockMode, contentChecksumFlag frameType and contentSize.

Return type dict

lz4.block sub-package

This sub-package provides the capability to compress and decompress data using the LZ4 block specification <http://lz4.github.io/lz4/lz4_Block_format.html>

Because the LZ4 block format doesn't define a container format, the Python bindings will by default insert the original data size as an integer at the start of the compressed payload, like most other bindings do (Java...). However, it is possible to disable this functionality.

Example usage

To use the lz4 block format bindings is straightforward:

```
>>> import lz4.block
>>> compressed_data = lz4.block.compress(data)
>>> data == lz4.block.decompress(compressed_data)
True
>>>
```

Contents

lz4.block.**compress**(*source*, *mode='default'*, *acceleration=1*, *compression=0*)
Compress source, returning the compressed data as a string. Raises an exception if any error occurs.

Parameters

- **source** (*str, bytes or buffer-compatible object*) – Data to compress
- **mode** (*str*) – If 'default' or unspecified use the default LZ4 compression mode. Set to 'fast' to use the fast compression LZ4 mode at the expense of compression. Set to 'high_compression' to use the LZ4 high-compression mode at the exepense of speed
- **acceleration** (*int*) – When mode is set to 'fast' this argument specifies the acceleration. The larger the acceleration, the faster the but the lower the compression. The default compression corresponds to a value of 1.
- **compression** (*int*) – When mode is set to *high_compression* this argument specifies the compression. Valid values are between 1 and 12. Values between 4-9 are recommended, and 9 is the default.
- **store_size** (*bool*) – If True (the default) then the size of the uncompressed data is stored at the start of the compressed block.
- **return_bytearray** (*bool*) – Python 3 only. If False (the default) then the function will return a bytes object. If True, then the function will return a bytearray object.

Returns Compressed data.

Return type bytes or bytearray

lz4.block.**decompress**(*source*, *uncompressed_size=-1*)
Decompress source, returning the uncompressed data as a string. Raises an exception if any error occurs.

Parameters

- **source** (*str, bytes or buffer-compatible object*) – Data to decompress
- **uncompressed_size** (*int*) – If not specified or < 0, the uncompressed data size is read from the start of the source block. If specified, it is assumed that the full source data is compressed data.
- **return_bytearray** (*bool*) – Python 3 only. If False (the default) then the function will return a bytes object. If True, then the function will return a bytearray object.

Returns Decompressed data.

Return type bytes or bytearray

Is it fast ?

Yes. Here are the results on my 2011 Macbook Pro i7 with lz4.c as input data:

```
$ python tests/bench.py
Data Size:
  Input: 24779
  LZ4: 10152 (0.41)
  Snappy: 9902 (0.40)
  LZ4 / Snappy: 1.025247
Benchmark: 200000 calls
  LZ4 Compression: 9.737272s
  Snappy Compression: 18.012336s
```

```
LZ4 Decompression: 2.686854s
Snappy Decompression : 5.146867s
```

Contributors

- Jonathan Underwood combined the block and frame modules into a coherent single project with many fixes, clean-ups and documentation
- Jonathan Underwood added frame bindings based on the lz4ex by Jerry Ryle and the lz4tools project by Christopher Jackson
- Jonathan Underwood updated the block format support to use the tunable accelerated and high compression functions
- Mathew Rocklin added support for dropping the GIL to the block module, and Travis testing support
- Antoine Martin added initial support for fast compression support to the block library
- Steve Morin wrote the original lz4 block bindings

Licensing

Code specific to this project is covered by the BSD 3-Clause License

CHAPTER 2

Indices

- genindex
- modindex
- search

l

C

D

G

L