インメモリーストリーム活用術

How to Use In-Memory Streams

Hayao Suzuki

PyCon JP 2020

August 29, 2020

発表に際して

GitHub に資料があります

- https://github.com/HayaoSuzuki/pyconjp2020

Twitter のハッシュタグ

- #pyconjp_1

PyCon JP Fellow Slack

- #jp-2020-track-1

Who am I ?

お前誰よ

Name	Hayao Suzuki（鈴木　駿）
Twitter	@CardinalXaro
Work	Python Programmer at iRidge, Inc.

Who am I ?

Technical Reviewer

- Effective Python 第 2 版 (O'Reilly Japan)
- 動かして学ぶ量子コンピュータプログラミング (O'Reilly Japan)

`https://xaro.hatenablog.jp/` にリストがあります。

Who am I ?

Selected Talks

- レガシー Django アプリケーションの現代化 (DjangoCongress JP 2018)
- SymPy による数式処理 (PyCon JP 2018)
- Python と楽しむ初等整数論 (PyCon mini Hiroshima 2019)
- 君は cmath を知っているか (PyCon mini Shizuoka 2020)

`https://xaro.hatenablog.jp/` にリストがあります。

今日の目標

こんな課題を解決したい！

- インターネット経由で数 GB サイズのデータを取得し、CSV ファイルに加工する
- クラウド上に構築した既存のシステムに追加する形で実装する
- 毎日実行する

クラウドサービスは従量課金

なるべく迅速に処理したい！

今日の目標

処理の流れ

- インターネット経由で数 GB サイズのデータを取得する
- 数 GB サイズのデータを CSV ファイルに加工する
- CSV ファイルを ZIP 圧縮する
- ZIP 圧縮データをクラウドストレージにアップロードする

分析

- データサイズが大きい
- データの加工は単純な処理

今日の目標

ボトルネックはどこか

- ZIP 圧縮はそれほど大変ではない
- データ加工は単純な処理
- ボトルネックは I/O 処理にありそう

何とかして I/O 処理を迅速に処理したい！！！

In-Memory Streams

Stream?

そもそもストリームって何？

ストリームはファイルオブジェクトである。

File Object?

ファイルオブジェクトって何？

- `read()` や `write()` などのメソッドを持つオブジェクト
- ディスク上のファイルや別の場所にあるストレージ、入出力機器とやりとりができる

File Object?

ファイルオブジェクトたち

- 生バイナリファイル
- バッファ付きバイナリファイル
- テキストファイル

使い方

テキストファイル

```
f = open("myfile.txt", "r")
```

バッファ付きバイナリ

```
f = open("myfile.jpg", "rb")
```

open 関数の裏側

open は何をしているのか？

OS のシステムコール API を呼ぶ

open 関数の裏側

例：CSV に加工する

```
with open("events.csv", "w") as csv_file:
    fieldnames = ["title", "started_at", "ended_at"]
    writer = csv.DictWriter(csv_file, fieldnames)
    writer.writeheader()
    writer.writerows(events)
```

open 関数の裏側

例：Windows

- CreateFile（ファイルのアクセス権取得）
- QueryAllInformationFile（ファイル情報の取得）
- WriteFile（ファイルへ書き込む）
- CloseFile（ファイルを閉じる）

Process Monitor 経由で確認した。

open 関数の裏側

例：Ubuntu on WSL

- openat （ファイルのオープン）
- fstat（ファイル情報の取得）
- ioctl（デバイス制御）
- lseek（ファイルのシーク）
- write（ファイルへ書き込む）
- close（ファイルを閉じる）

`strace` 経由で確認した。

最後に笑うのは誰だ

最終的な成果物はどこに置く？

- ファイルをローカルに保存するのがゴールではない
- ファイルを AWS S3 などの外部に置きたい

ローカルデバイスにファイルを書き込みたくない！

In-Memory Streams

インメモリーストリーム

インメモリーストリームとは

- `str` や `bytes` をファイルオブジェクトのように扱える
- 読み書き可能、ランダムアクセス可能

StringIO

StringIO

テキストファイルのためのインメモリストリーム

例：CSV を StringIO で取り扱う

```
import io
with io.StringIO() as csv_file:
    fieldnames = ["title", "started_at", "ended_at"]
    writer = csv.DictWriter(csv_file, fieldnames)
    writer.writeheader()
    writer.writerows(events)
```

BytesIO

BytesIO

バッファ付きバイナリファイルのためのインメモリストリーム

例：PNG を BytesIO で取り扱う

```
import io
with io.BytesIO(png_bytes) as f:
    png_header = f.read(8)
    print(png_header)  # b'\x89PNG\r\n\x1a\n'
```

復習：今日の目標

処理の流れ

- インターネット経由で数 GB サイズのデータを取得する
- 数 GB サイズのデータを CSV ファイルに加工する
- CSV ファイルを ZIP 圧縮する
- ZIP 圧縮データをクラウドストレージにアップロードする

データをインターネット経由で取得する

例：Connpass API をコールする

```
with urllib.request.urlopen(url) as response:
    events = json.load(response)["events"]
```

データを加工する

例：API の取得結果を CSV にする

```
with io.StringIO() as ts:
    header = ["title", "started_at", "ended_at"]
    writer = csv.DictWriter(ts, fieldnames=header)
    writer.writeheader()
    writer.writerows(events)
```

データを圧縮&アップロード

例：ZIP に圧縮して AWS S3 にアップロード

```
with io.BytesIO() as bs:
    with zipfile.ZipFile(bytes_stream, "w") as zf:
        zf.writestr("events.csv", ts.getvalue())
    bs.seek(0)  # ファイルシークがポイント
    s3.upload_fileobj(bs, "bucket", "events.zip")
```

Conclusion

まとめ

- io モジュールにはインメモリーストリームが含まれる。
- str や bytes をファイルオブジェクトのように扱うことができる。
- 通常の open と異なりシステムコールが呼ばれない。
- ディスクへの I/O を減らしたい、またはできない状況下での利用が最適である。

io モジュールを皆様の道具箱に入れてください！