巨大なデータを保存・復元する

～ なぜ、あなたのPickleは本気を出さないのか？～

機械学習の結果など

```
[
    {'id1': ['data1': 0.65442,
             'data2': 1.32543,
             ...
            ]
    },
    {'id2': ['data1': -3.15776,
             'data2': 5.32543,
             ...
            ]
    },
    {'id3': ['data1': 6.23147,
             'data2': 0.44531,
             ...
            ]
    }, ...
]
```

よくPickleで保存する

```
import pickle
with open("large_data.pkl", "wb") as f:
    pickle.dump(data, f)
```

巨大なPickleファイルの読み込むとき、

```
import pickle
with open("large_data.pkl", "rb") as f:
    data = pickle.load(f)
```

速度がちょっと...

```
178.57 sec
```

さらには、

散々待たされて、メモリエラー...

```
*** set a breakpoint in malloc_error_break to debug
python(3716,0xa08ed1d4) malloc: *** mach_vm_map(size=1048576) f
*** error: can't allocate region securely
*** set a breakpoint in malloc_error_break to debug
Traceback (most recent call last):
  File "/System/Library/Frameworks/Python.framework/Versions/2.
    dispatch[key](self)
  File "/System/Library/Frameworks/Python.framework/Versions/2.
    self.stack.append({})
MemoryError
```

そもそもPickleってなに？

Pickleとは

- オブジェクトをバイト列に変換する方法（Serialization）
  - 例：dict型の変数 -> バイナリファイル
  - 例：定義したクラスインスタンス -> バイナリファイル
- 直接ファイルを開いても読めない
- Pythonにしかない
- 機械学習モデルの保存によく使われる

Pickleを早くする方法を調べました

Pickleを早くする方法５つ

- `cPickle` モジュールを使う
- `protocol=4` を使う
- `fast=True` を使う
- `pickletools.optimize()` を使う
- generatorを使う

1つ目：`cPickle`

- 標準のpickleをC言語で書き直したやつ
  - `pickle`：Pythonで書いてある
  - `cPickle`：C言語で書いてある
- 使い方

```
import pickle
```

↓

```
import cPickle # Python2系
import _pickle # Python3系
```

2つ目：`protocol=4`

- pickleモジュールの最新プロトコルを使う
  - `protocol=0`: python2系のデフォルト
  - `protocol=3`: python3系のデフォルト、2系では使えない
  - `protocol=4`: 巨大オブジェクトをサポート
- 使い方

```
pickle.dump(x, f)
```

↓

```
pickle.Pickler(f, protocol=4).dump(x)
```

3つ目：`fast=True`

- 余分な PUT 命令コードを生成しなくなる
- 公式によると廃止予定
- 使い方

```
pickle.dump(data)
```

↓

```
p = pickle.Pickler(f)
p.fast = True # ここ
p.dump(data)
```

4つ目：`pickletools.optimize()`

- 余分な PUT 命令コードを生成しなくなる（`fast=True` と同じ）
- 使い方

```
pickle.dump(data)
```

↓

```
import pickletools

pickled = pickle.dumps(data)
opt = pickletools.optimize(pickled)
pickle.dump(opt, f)
```

5つ目：generator

- リスト全体を保持しないため、メモリ消費が少ない
    - 次の要素だけ参照できる
- やり方

```
data_list = [i for i in range(500000)] # ふつうのリスト
pickle.dump(data_list, f)
```

↓

```
data_generator = (i for i in range(500000)) # Generator
for x in data_generator:
    pickle.dump(x, f)
```

比較してみた

比較するPickle一覧

		オプションなし	fast=True	pickletools.optimize()
list	pickle	pickle0	pickle0_fast	pickle0_opt
		pickle3	pickle3_fast	pickle3_opt
		pickle4	pickle4_fast	pickle4_opt
	cPickle	cPickle3	cPickle3_fast	cPickle3_opt
		cPickle4	cPickle4_fast	cPickle4_opt
generator	pickle	pickle0_gen	pickle0_gen_fast	pickle0_gen_opt
		pickle3_gen	pickle3_gen_fast	pickle3_gen_opt
		pickle4_gen	pickle4_gen_fast	pickle4_gen_opt
	cPickle	cPickle3_gen	cPickle3_gen_fast	cPickle3_gen_opt
		cPickle4_gen	cPickle4_gen_fast	cPickle4_gen_opt

比較項目

- 時間
  - `dump` にかかる時間
  - `load` にかかる時間
- 最大メモリ使用量
  - `dump` に使うメモリ
  - `load` に使うメモリ
- `dump` されたファイルサイズ

`dump` / `load` するデータ（リスト版）

大体 1 GBのデータ、100列 x 500000行くらい

```
[
    {
     'id': 1,
     'data': ['data1.1', 'data1.2', ..., 'data1.100']
    },
    ...
    {
     'id': 500000,
     'data': ['data500000.1', ..., 'data500000.100']
    }
]
```

`dump` / `load` するデータ（Generator版）

大体 1 GBのデータ、100列 x 500000行くらい

```
{
 'id': 1,
 'data': ['data1.1', 'data1.2', ..., 'data1.100']
},
...
{
 'id': 500000,
 'data': ['data500000.1', ..., 'data500000.100']
}
```

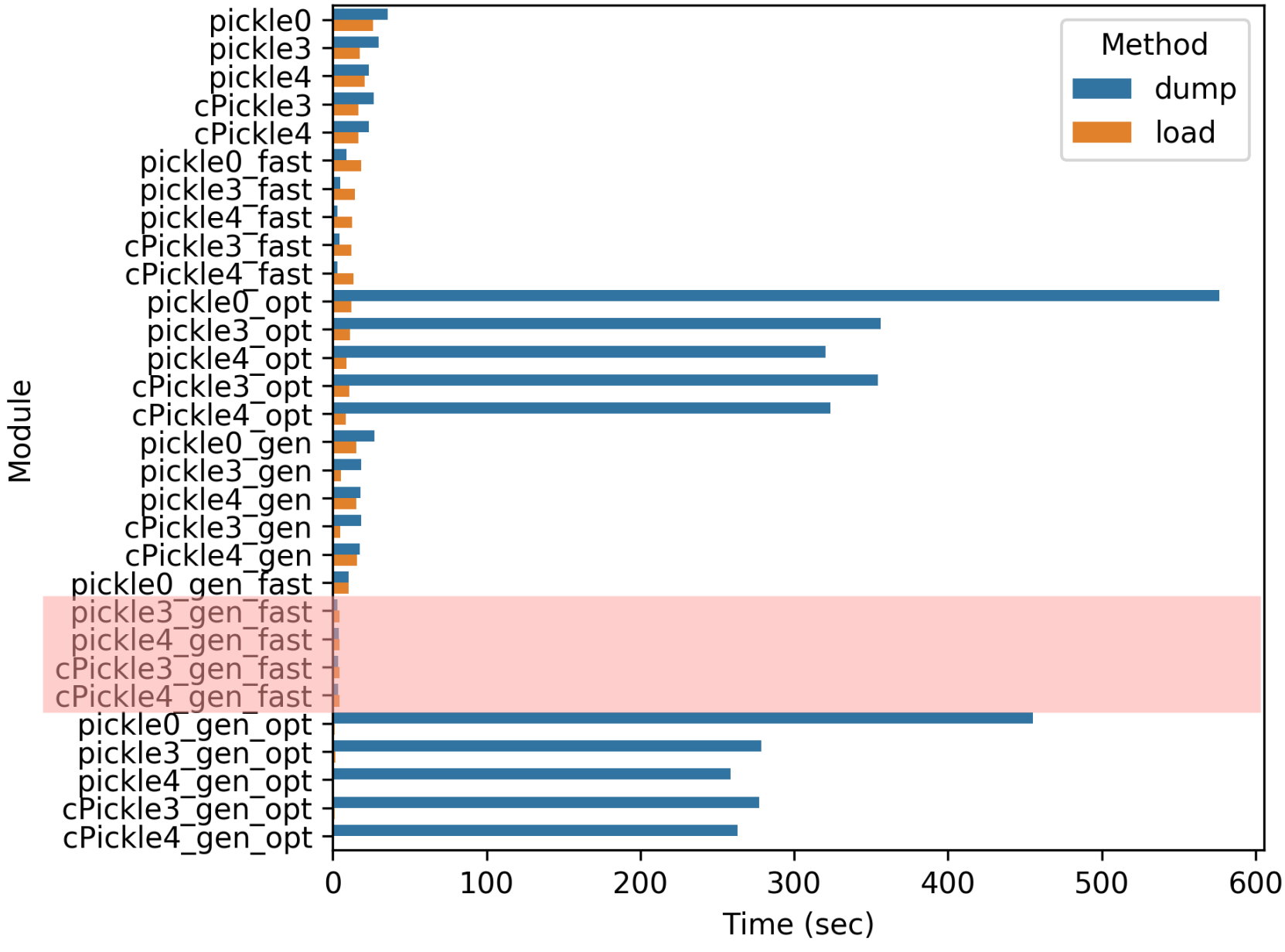
環境

- Python 3.7.2
- MacBook Pro (2017)

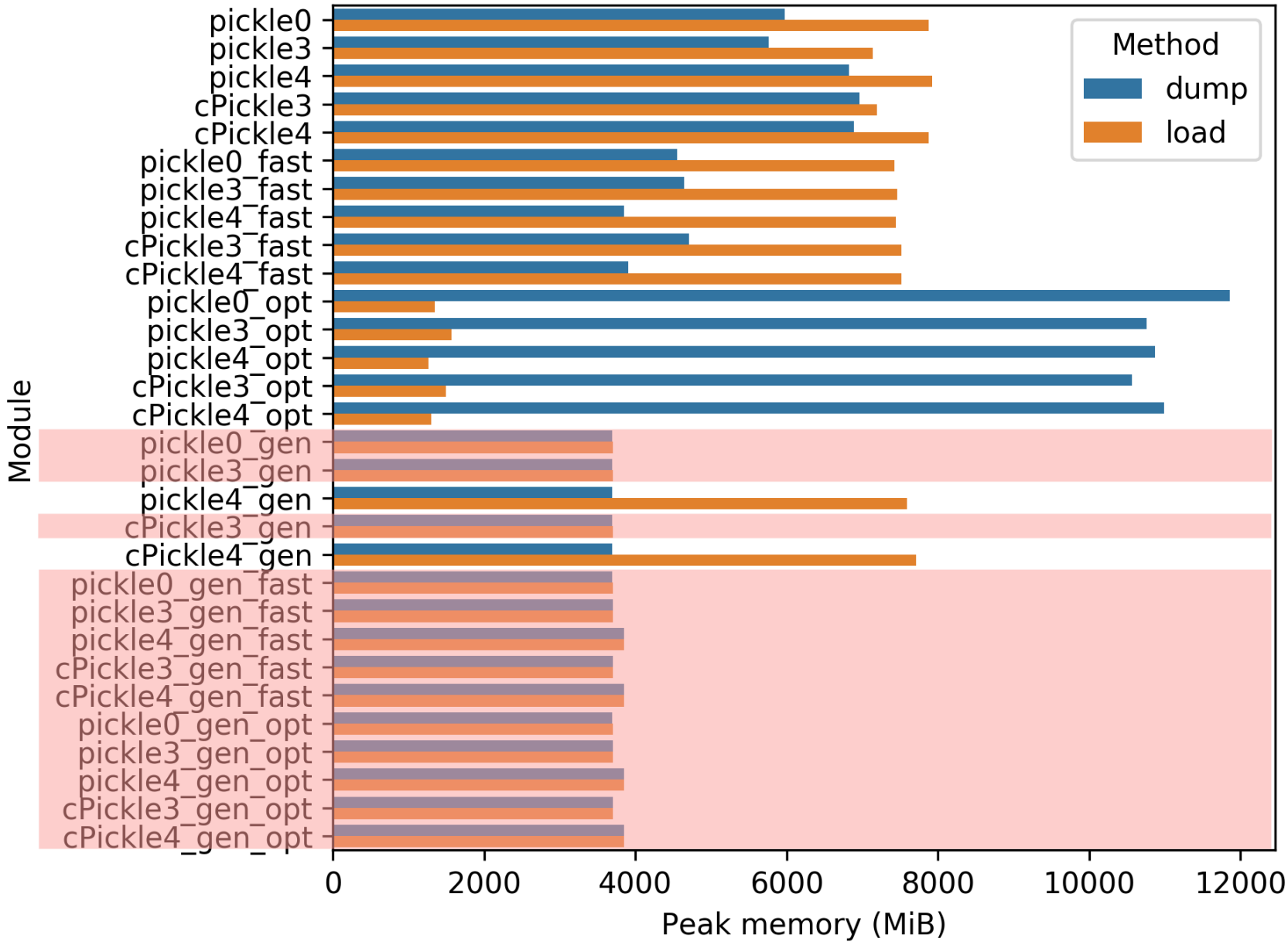
```
プロセッサ: 2.3 GHz Intel Core i5
メモリ: 16 GB 2133 MHz LPDDR3
```

結果

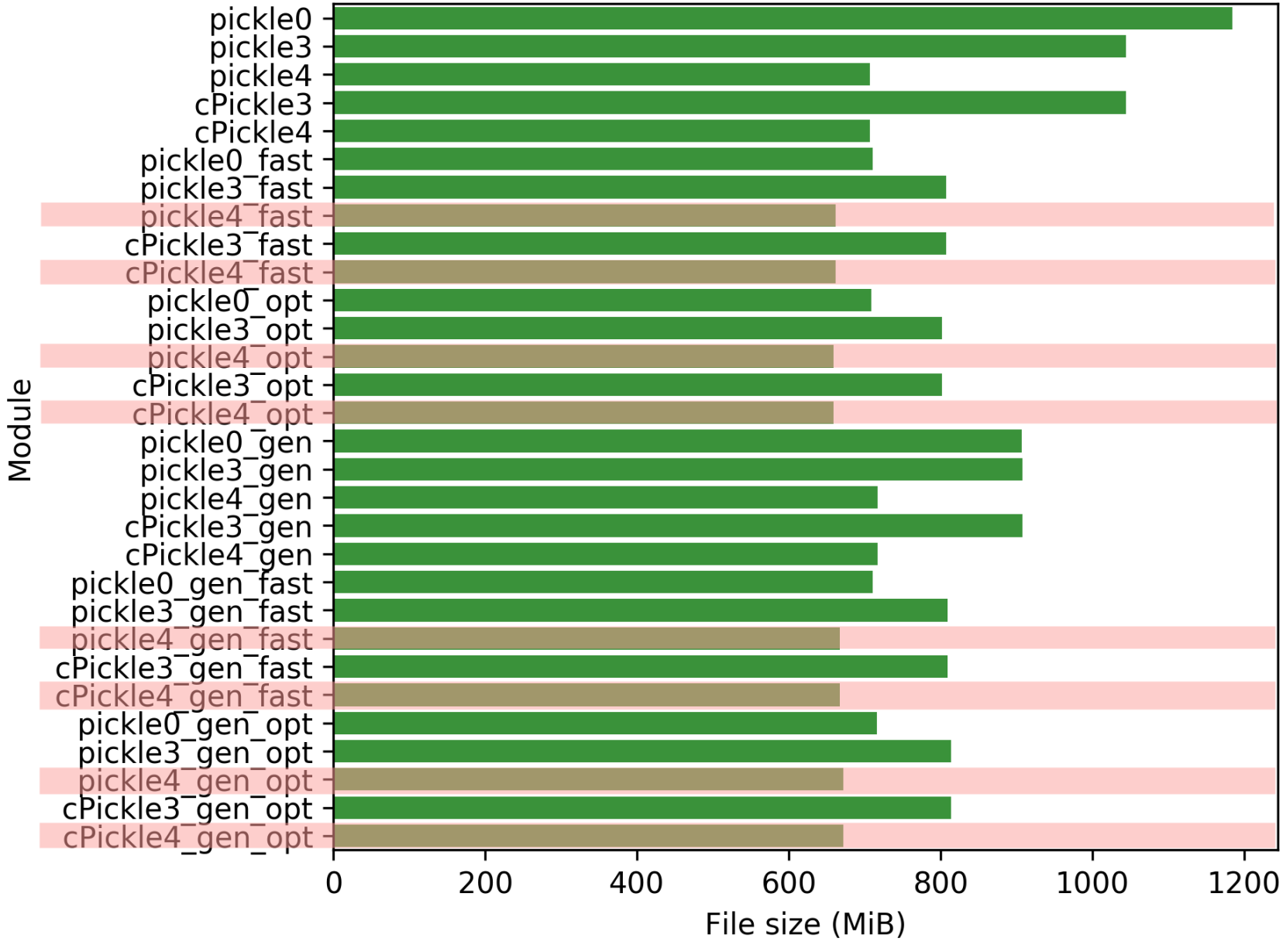
時間


Method
dump
load
Module
pickle0
pickle3
pickle4
cPickle3
cPickle4
pickle0_fast
pickle3_fast
pickle4_fast
cPickle3_fast
cPickle4_fast
pickle0_opt
pickle3_opt
pickle4_opt
cPickle3_opt
cPickle4_opt
pickle0_gen
pickle3_gen
pickle4_gen
cPickle3_gen
cPickle4_gen
pickle0_gen_fast
pickle3_gen_fast
pickle4_gen_fast
cPickle3_gen_fast
cPickle4_gen_fast
pickle0_gen_opt
pickle3_gen_opt
pickle4_gen_opt
cPickle3_gen_opt
cPickle4_gen_opt
0
100
200
300
400
500
600
Time (sec)

最大メモリ使用量

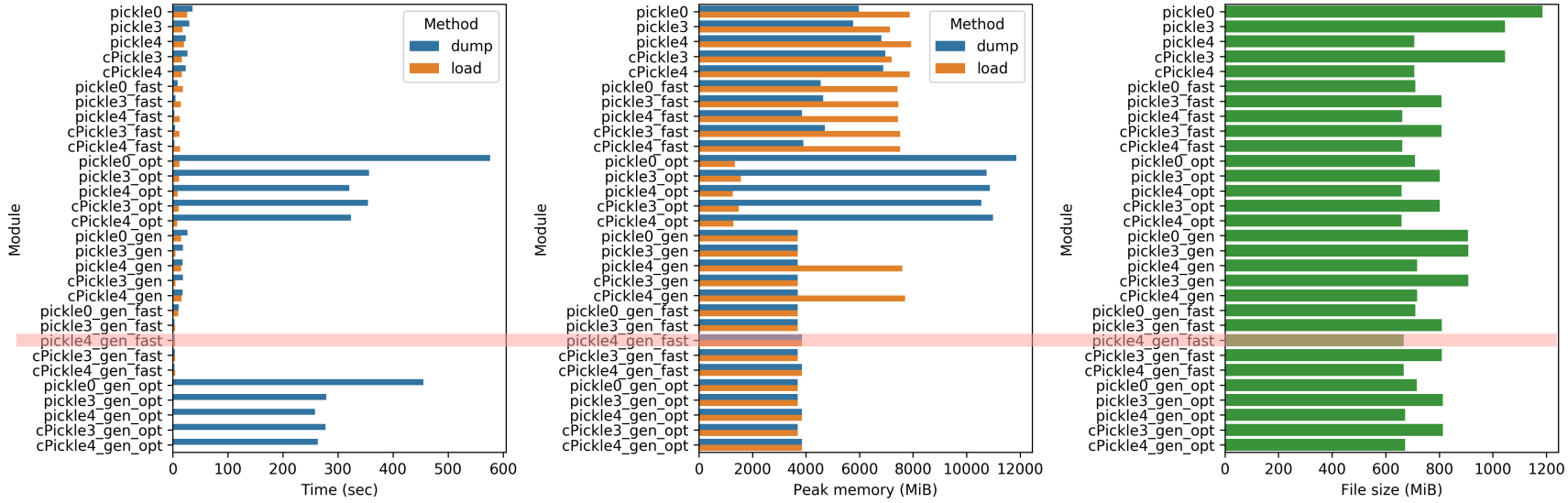

Method
dump
load
Module
pickle0
pickle3
pickle4
cPickle3
cPickle4
pickle0_fast
pickle3_fast
pickle4_fast
cPickle3_fast
cPickle4_fast
pickle0_opt
pickle3_opt
pickle4_opt
cPickle3_opt
cPickle4_opt
pickle0_gen
pickle3_gen
pickle4_gen
cPickle3_gen
cPickle4_gen
pickle0_gen_fast
pickle3_gen_fast
pickle4_gen_fast
cPickle3_gen_fast
cPickle4_gen_fast
pickle0_gen_opt
pickle3_gen_opt
pickle4_gen_opt
cPickle3_gen_opt
cPickle4_gen_opt
0
2000
4000
6000
8000
10000
12000
Peak memory (MiB)

ファイルサイズ


pickle0
pickle3
pickle4
cPickle3
cPickle4
pickle0_fast
pickle3_fast
pickle4_fast
cPickle3_fast
cPickle4_fast
pickle0_opt
pickle3_opt
pickle4_opt
cPickle3_opt
cPickle4_opt
pickle0_gen
pickle3_gen
pickle4_gen
cPickle3_gen
cPickle4_gen
pickle0_gen_fast
pickle3_gen_fast
pickle4_gen_fast
cPickle3_gen_fast
cPickle4_gen_fast
pickle0_gen_opt
pickle3_gen_opt
pickle4_gen_opt
cPickle3_gen_opt
cPickle4_gen_opt
Module
0
200
400
600
800
1000
1200
File size (MiB)

総合優勝

pickle4_gen_fast



まとめ

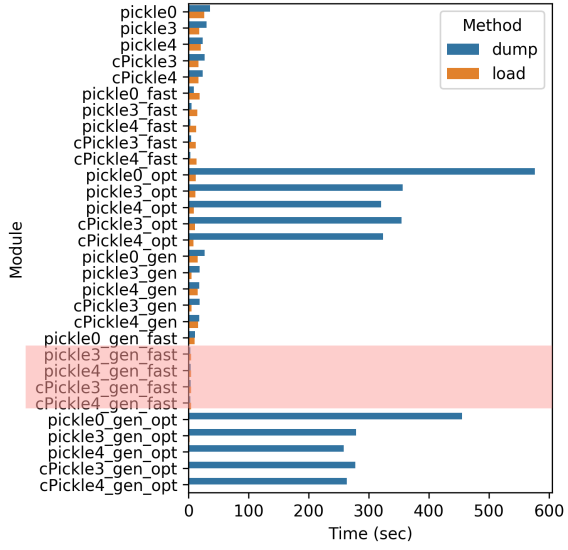
本気を出したPickleの書き方

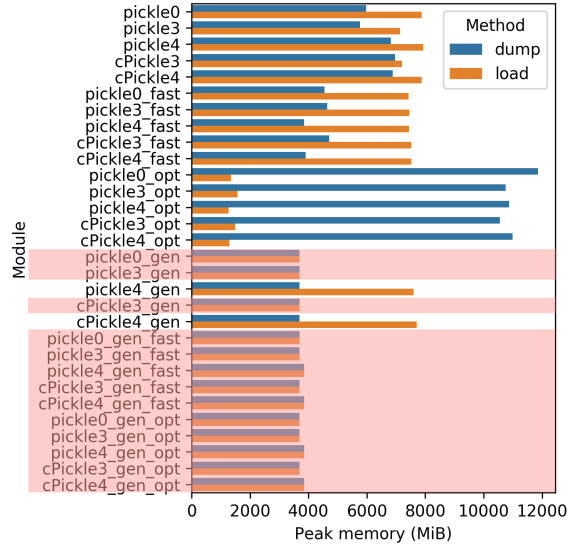
```
import pickle

p = pickle.Pickler(f, protocol=4)
p.fast = True
for x in data_generator:
    p.dump(x)
```

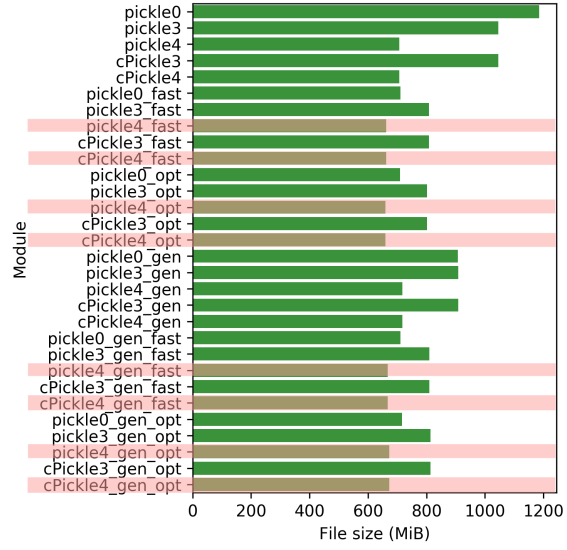
- `load` と `dump` かなり早い
- メモリふつう
- ファイルサイズ最小

- **ただし廃止予定**

おまけ


Method
dump
load
Module
pickle0
pickle3
pickle4
cPickle3
cPickle4
pickle0_fast
pickle3_fast
pickle4_fast
cPickle3_fast
cPickle4_fast
pickle0_opt
pickle3_opt
pickle4_opt
cPickle3_opt
cPickle4_opt
pickle0_gen
pickle3_gen
pickle4_gen
cPickle3_gen
cPickle4_gen
pickle0_gen_fast
pickle3_gen_fast
pickle4_gen_fast
cPickle3_gen_fast
cPickle4_gen_fast
pickle0_gen_opt
pickle3_gen_opt
pickle4_gen_opt
cPickle3_gen_opt
cPickle4_gen_opt
0
100
200
300
400
500
600
Time (sec)


Method
dump
load
Module
pickle0
pickle3
pickle4
cPickle3
cPickle4
pickle0_fast
pickle3_fast
pickle4_fast
cPickle3_fast
cPickle4_fast
pickle0_opt
pickle3_opt
pickle4_opt
cPickle3_opt
cPickle4_opt
pickle0_gen
pickle3_gen
pickle4_gen
cPickle3_gen
cPickle4_gen
pickle0_gen_fast
pickle3_gen_fast
pickle4_gen_fast
cPickle3_gen_fast
cPickle4_gen_fast
pickle0_gen_opt
pickle3_gen_opt
pickle4_gen_opt
cPickle3_gen_opt
cPickle4_gen_opt
0
2000
4000
6000
8000
10000
12000
Peak memory (MiB)


Module
pickle0
pickle3
pickle4
cPickle3
cPickle4
pickle0_fast
pickle3_fast
pickle4_fast
cPickle3_fast
cPickle4_fast
pickle0_opt
pickle3_opt
pickle4_opt
cPickle3_opt
cPickle4_opt
pickle0_gen
pickle3_gen
pickle4_gen
cPickle3_gen
cPickle4_gen
pickle0_gen_fast
pickle3_gen_fast
pickle4_gen_fast
cPickle3_gen_fast
cPickle4_gen_fast
pickle0_gen_opt
pickle3_gen_opt
pickle4_gen_opt
cPickle3_gen_opt
cPickle4_gen_opt
0
200
400
600
800
1000
1200
File size (MiB)

次回予告（未定）

巨大なデータを保存・復元する

～ あなたの知らないSerializationの世界 ～

```
pickle
cPickle
json
csv
msgpack
hdfstore
marshal
dill
cloudpickle
hickle
...
```