GTC Japan 2018

CuPy

NumPy互換GPUライブラリによるPythonでの高速計算

Preferred Networks
取締役　最高技術責任者　奥田遼介 okuta@preferred.jp

CuPyとは何か？

CuPyとは

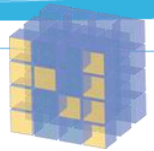
GPUを使ってNumPy互換の機能を提供するライブラリ


```
import numpy as np
X_cpu = np.zeros((10,))
W_cpu = np.zeros((10, 5))
y_cpu = np.dot(x_cpu, W_cpu)
```

```
import cupy as cp
x_gpu = cp.zeros((10,))
W_gpu = cp.zeros((10, 5))
y_gpu = cp.dot(x_gpu, W_gpu)
```

```
y_cpu = cp.asnumpy(y_gpu)
```

```
y_gpu = cp.asarray(y_cpu)
```



NumPy

CuPy

```
import numpy as np
X_cpu = np.zeros((10,))
W_cpu = np.zeros((10, 5))
y_cpu = np.dot(x_cpu, W_cpu)
```

```
import cupy as cp
x_gpu = cp.zeros((10,))
W_gpu = cp.zeros((10, 5))
y_gpu = cp.dot(x_gpu, W_gpu)
```

```
for xp in [np, cp]:
    x = xp.zeros((10,))
    W = xp.zeros((10, 5))
    y = xp.dot(x, W)
```

CuPyにより一つのコードでCPU/GPUをサポート

なぜCuPyを作ったのか？（その1）

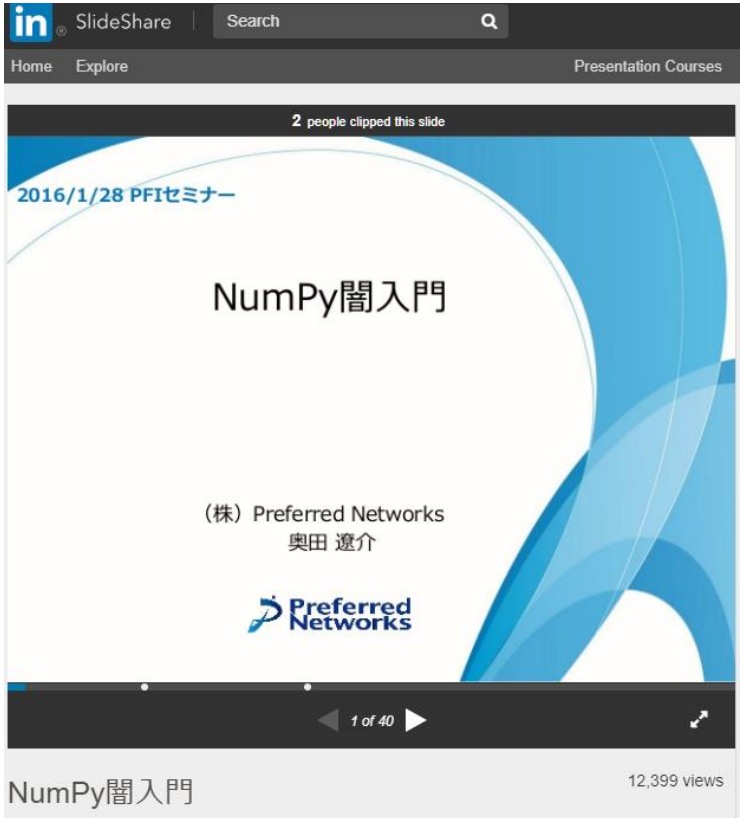
- Chainerの関数を書くときにNumPyとPyCUDA 両方のコードを書いていた

AddとかConcatとか
シンプルな関数を
ぱっとかけない辛さ

```
7   _args = 'const float* x, float* y, int cdimx, int cdimy, int rdim, int coffset'
8   _preamble = '''
9   #define COPY(statement) \
10      int l   = i / (rdim * cdimx);  \
11      int c   = i / rdim % cdimx + coffset;  \
12      int r   = i % rdim;  \
13      int idx = r + rdim * (c + cdimy * l);  \
14      statement;
15  '''
16
17
18  class Concat(function.Function):
19
20      """Concatenate multiple tensors towards specified axis."""
```

なぜCuPyを作ったのか？（その2）

- NumPyと高い互換性を持つことが必要
  - dtype, Broadcast, Indxing, バグ・・・
- NumPy闇入門
  - それらの調査の成果物⇒



https://www.slideshare.net/ryokuta/numpy-57587130

なぜCuPyを作ったのか？（その3）

- そんな都合のいいライブラリが無かった
  - gnumpy
    - 約1000行のシングルファイル！ライブラリ
  - CUDA-based NumPy
    - pip packageが無い

⇒自分たちで開発する必要性に気づく

ChainerのためのバックエンドとしてCuPy誕生

CuPy: Add and use a new GPU array backend with NumPy-compatible interface #266

Edit

Merged **beam2d** merged 305 commits into `master` from `cupy` on 20 Aug 2015

Conversation 6 | Commits 250+ | Checks 0 | Files changed 297 | +15,915 −4,904

beam2d commented on 27 Jul 2015 — Member

This is a large PR aiming at replacing the CUDA array backend from PyCUDA/scikit-cuda to a new one named *CuPy*. This PR includes the implementation of CuPy and updates on Chainer.

Background: PyCUDA is a great wrapper of CUDA that enables us to write our own kernels and call them from Python. However, its GPUArray has few functionalities and almost every time we have to write our own kernels to write down Function implementations. We want to make it easier to write user-defined Functions runnable on GPU. It requires us to use more powerful GPU-array implementations.

Reviewers
No reviews

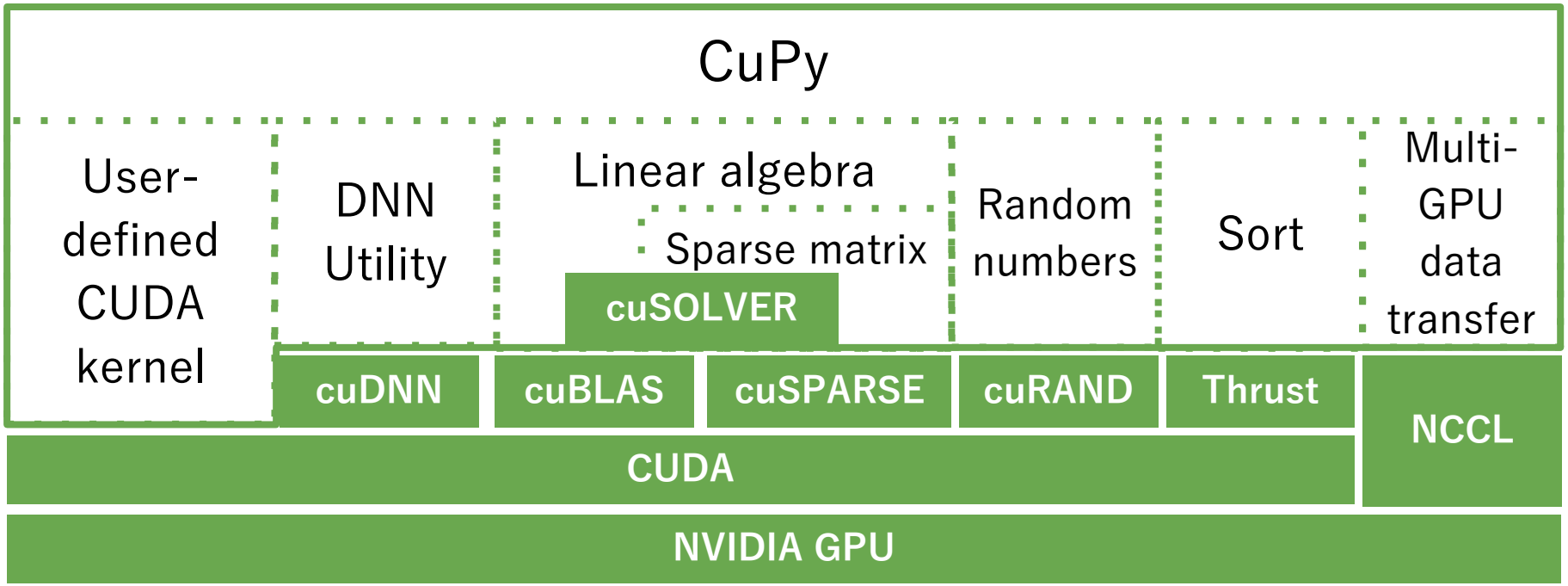
Assignees
No one—assign yourself

Labels
feature

CuPyの歴史

2015/6/5	Chainer v1.0	PyCUDA時代
2015/7/?		CuPy開発開始
2015/9/2	Chainer v1.3	PyCUDAからCuPyへ
2017/2/21	CuPy v1.0 a1	CuPy独立
2018/4/17	CuPy v4.0	毎月1回のリリース体制へ

Inside CuPy


CuPy
User-defined CUDA kernel
DNN Utility
Linear algebra
Sparse matrix
cuSOLVER
Random numbers
Sort
Multi-GPU data transfer
cuDNN
cuBLAS
cuSPARSE
cuRAND
Thrust
NCCL
CUDA
NVIDIA GPU

NumPyとの互換機能一覧

- Data types (dtypes)
  - bool_, int8, int16, int32, int64, uint8, uint16,
    uint32, uint64, float16, float32, float64, complex64, and complex128
- All basic indexing
  - indexing by ints, slices, newaxes, and Ellipsis
- Most of advanced indexing
  - except indexing patterns with boolean masks
- Most of the array creation routines
  - empty, ones_like, diag, etc...
- Most of the array manipulation routines
  - reshape, rollaxis, concatenate, etc...
- All operators with broadcasting
- All universal functions for element-wise operations
  - except those for complex numbers
- Linear algebra functions accelerated by cuBLAS
  - including product: dot, matmul, etc...
  - including decomposition: cholesky, svd, etc...
- Reduction along axes
  - sum, max, argmax, etc...
- Sort operations implemented by Thrust
  - sort, argsort, and lexsort
- Sparse matrix accelerated by cuSPARSE

CuPy v2以降の取り組み

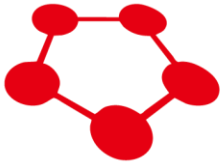
- NumPyとの差分の改善
- 速度向上：Cython化、MemoryPoolの改善
- CUDA Stream サポート
- 対応関数の充実
  - NumPy
  - Sparse Matrix, FFT, scipy ndimage 対応

他のライブラリとの比較

	CuPy	PyCUDA*	Theano	MinPy**
NVIDIA CUDA support	✔	✔	✔	✔
CPU/GPU agnostic coding	✔		✔	✔
Autograd support	***		✔	✔
NumPy compatible Interface	✔			✔
User-defined CUDA kernel	✔	✔		
				2018/2 開発終了

* https://github.com/inducer/pycuda
** https://github.com/dmlc/minpy
*** Autograd is supported by Chainer, a DL framework on top of CuPy

CuPyを活用したプロジェクト



Chainer

Deep learning framework
https://chainer.org/

pomegranate

Probabilistic and graphical modeling
https://github.com/jmschrei/pomegranate

spaCy

Natural language processing
https://spacy.io/

CuPyのOpenCL版実装CIPy

Fixstars Tech Blog /proc/cpuinfo

2018年8月23日

CuPyをOpenCLで動かすフレームワーク ClPy がベータ版になりました

投稿者：YOSHIFUJI Naoki、投稿日時：2018年8月23日14時27分

本日、フィックスターズは、CuPyをOpenCLでも使えるようにした「CIPy」のベータ版であるv2.1.0beta0を公開しました！

Search this site... Search

About us

FIXSTARS® Speed up your Business

このブログは、株式会社フィックスターズのエンジニアが、あらゆるテーマについて自由に書いているブログです。

CuPyの目指す方向

- **最小限の修正**でPythonで書いたコードをGPU対応にする
  - CPU向けライブラリとの高い互換性の確保
  - NumPyだけで無くSciPyなどにも対応

- **気軽にGPU**での高速化の検討が出来るようにする
  - インストールの簡易化
  - デフォルトで性能が出る設計

CuPyを使う

CuPyのインストール方法

1. CUDA SDKをインストールする
   - 必要ならcuDNN・NCCLをインストール
2. (環境変数 CUDA_PATHを設定)
   - 通常はSetupスクリプトが**自動で**CUDAを探します
3. **$ pip install cupy**

https://github.com/cupy/cupy#installation

高速にインストール出来るパッケージ

$ pip install cupy-cuda80　　(Binary Package for CUDA 8.0)
$ pip install cupy-cuda90　　(Binary Package for CUDA 9.0)
$ pip install cupy-cuda91　　(Binary Package for CUDA 9.1)
$ pip install cupy-cuda92　　(Binary Package for CUDA 9.2)

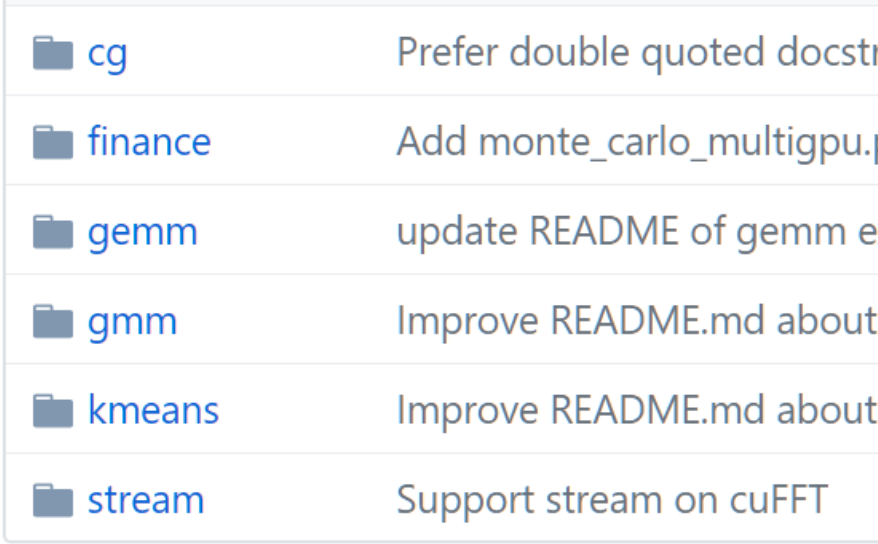
cuDNNとNCCLを同梱
（注：サポートしている環境に）

サンプル

```
import numpy as np
import cupy as cp
x_cpu = np.zeros((10, 10))

x_gpu = cp.asarray(x_cpu)        # copy CPU to GPU
x_cpu = cp.asnumpy(x_gpu)        # copy GPU to CPU

print(x_gpu ** 2)                # square on GPU by basic math

xp = cp.get_array_module(x_gpu) # get `np` or `cp`
print(xp.square(x_gpu))          # square on GPU by CuPy func
```

Examples
https://github.com/cupy/cupy/tree/master/examples

- CG法（共益勾配法）
- 金融（モンテカルロ法）
- 行列積（Raw Kernel）
- 混合ガウスモデル
- クラスタリング（K-means）
- CUDA Stream

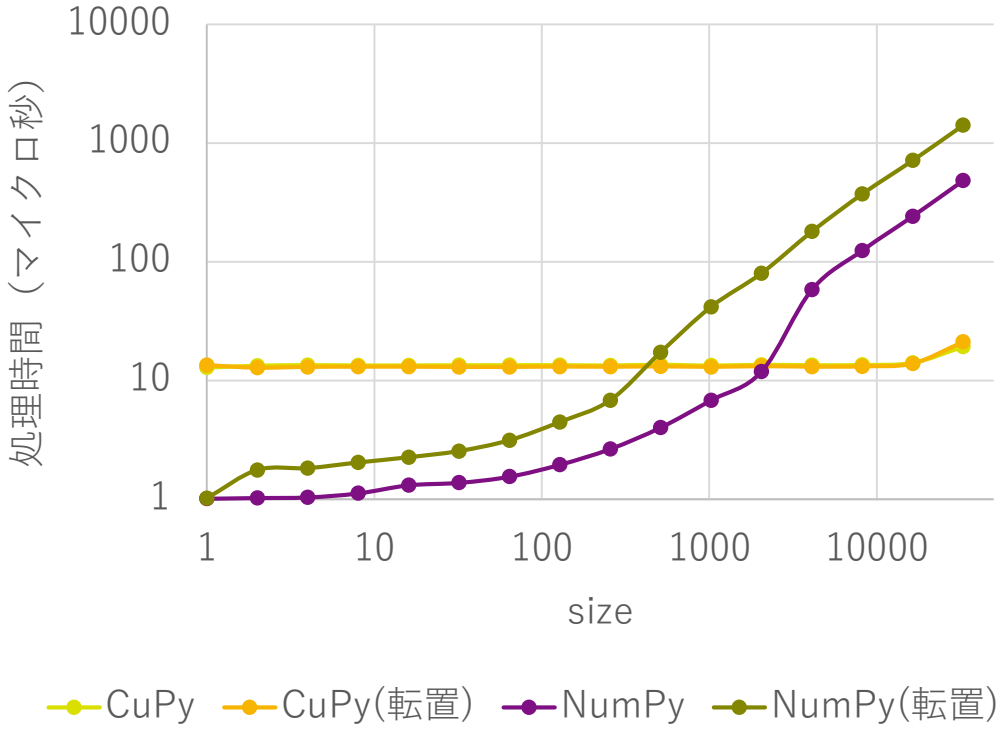


速度ベンチマーク

CuPyはどのくらい速くなるのか？（加算）

```
a = xp.ones((size, 32), 'f')
b = xp.ones((size, 32), 'f')

def f():
  a + b
```

```
# 転置
a = xp.ones((32, size), 'f').T
b = xp.ones((size, 32), 'f')

def f():
  a + b
```

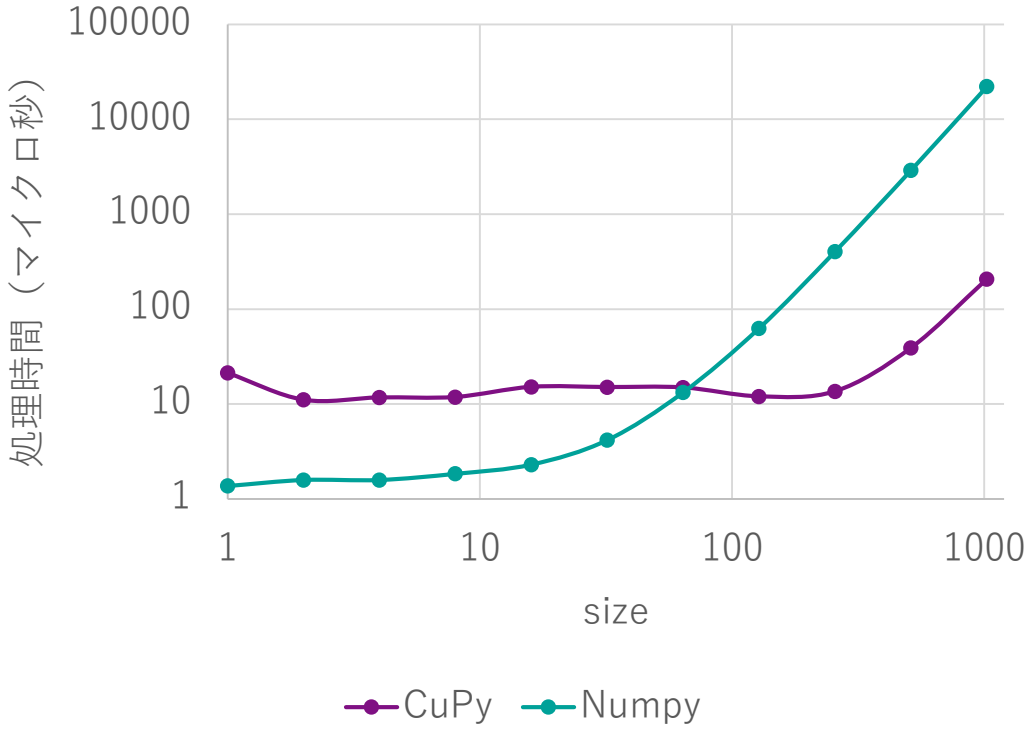


https://github.pfidev.jp/okuta/cupy-bench
Xeon Gold 6154 CPU @ 3.00GHz
Tesla V100-PCIE-16GB

CuPyはどのくらい速くなるのか？（内積）

```
a = xp.ones((size, size), 'f')
b = xp.ones((size, size), 'f')

def f():
  xp.dot(a, b)
```

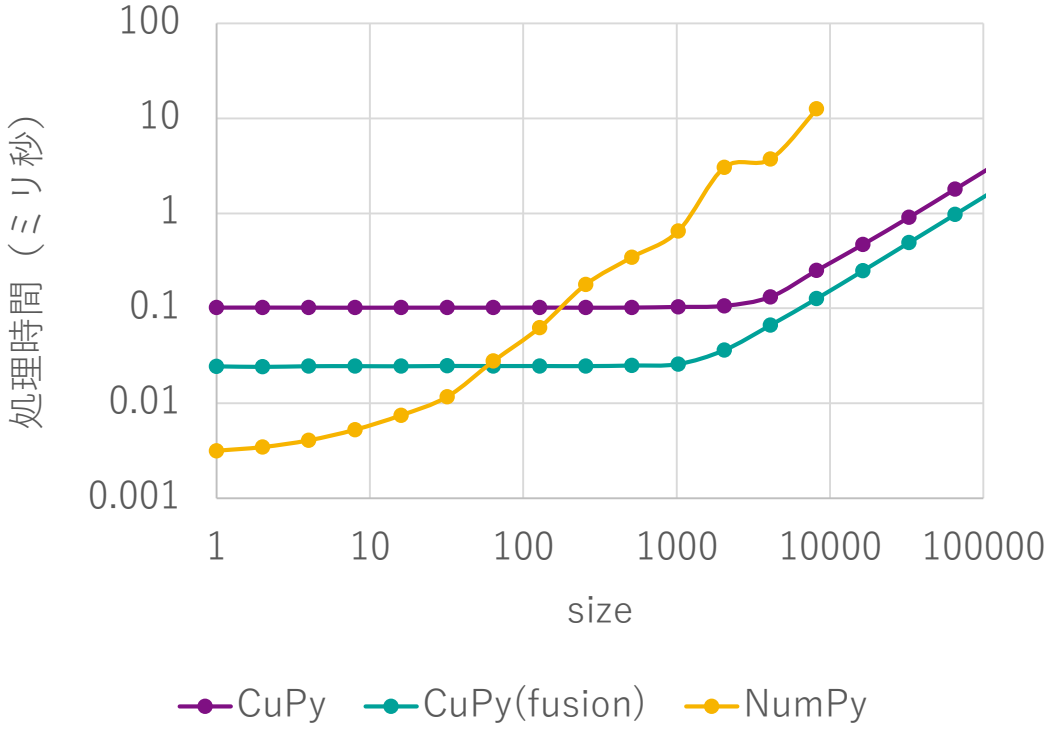
目安としてL1$-L2$に収まらない
サイズの計算はCuPyの方が速い



Fusionを活用した高速化

```
a = numpy.float32(2.0)
x = xp.ones((1024, size), 'f')
y = xp.ones((1024, size), 'f')

def saxpy(a, x, y):
    return a * x + y

saxpy(a, x, y)  # target
```

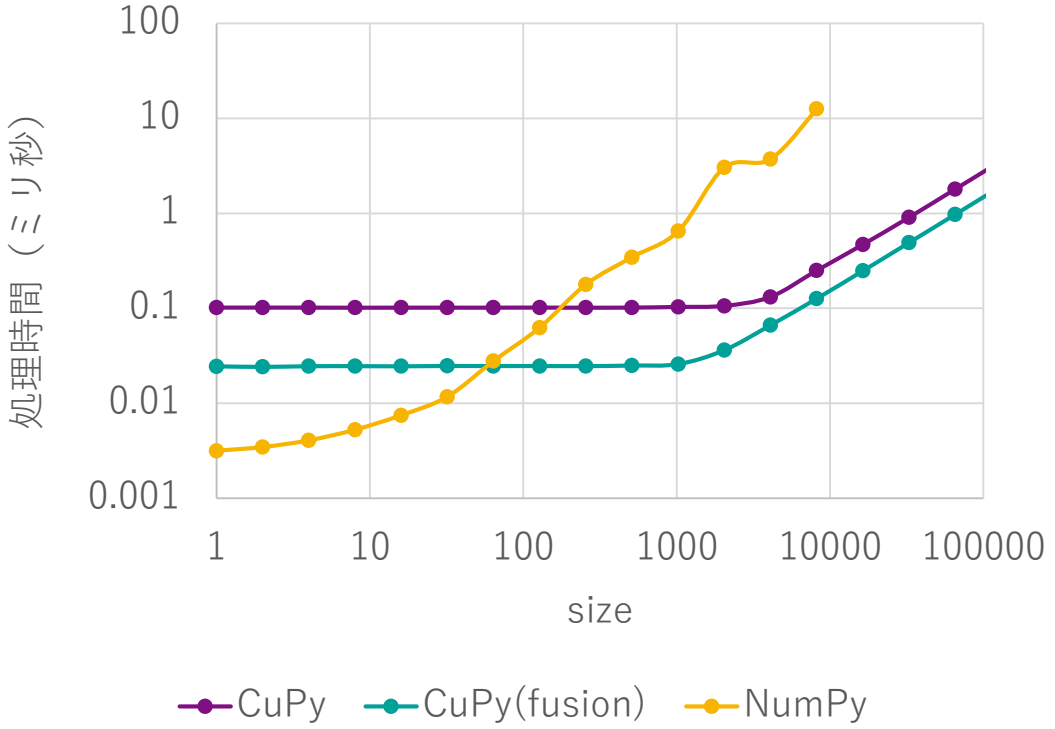
```
@cupy.fuse()
def saxpy(a, x, y):
    return a * x + y

saxpy(a, x, y) # target
```



Fusionを活用した高速化

Fusionのメリット

- 関数呼び出しを高速化
- メモリ使用量の削減
- 帯域律速の改善




```
@cupy.fuse()
def saxpy(a, x, y):
    return a * x + y

saxpy(a, x, y) # target
```

- GPUメモリの上限超える（Unified Memory）
- ユーザー定義カーネル
- Numbaとの連携

Advancedな機能の紹介

GPUメモリが足りない。そんな経験ありませんか？

```
import cupy as cp
size = 32768
a = cp.ones((size, size)) # 8GB
b = cp.ones((size, size)) # 8GB
cp.dot(a, b)              # 8GB
```

```
Traceback (most recent call last):
  ...
cupy.cuda.memory.OutOfMemoryError: out of memory to
allocate 8589934592 bytes (total 17179869184 bytes)
```

CuPy ＋Tesla V100なら簡単解決

- たった2行でUnifed Memoryを利用可能

```
import cupy as cp

pool = cp.cuda.MemoryPool(cp.cuda.malloc_managed)
cp.cuda.set_allocator(pool.malloc)

size = 32768
a = cp.ones((size, size)) # 8GB
b = cp.ones((size, size)) # 8GB
cp.dot(a, b)              # 8GB
```

ユーザー定義カーネル

- どうしてもCUDAを書きたいとき

- ElementwiseKernel
- ReductionKernel
- **RawKernel （v5）**
  - **自力で全部のコードを書くカーネル**

[v5] RawKernel のサンプル

```
import cupy as cp
square_kernel = cp.RawKernel(r'''
extern "C" __global__ void my_square(long long* x) {
    int tid = threadIdx.x;
    x[tid] *= x[tid];
}
''', 'my_square')

x = cp.arange(5)
square_kernel((1,), (5,), (x,)) # grid, block and arguments
print(x)                        # [ 0  1  4  9 16]
```

詳しいExampleあります https://github.com/cupy/cupy/tree/master/examples/gemm

CUDAのコードをどうしても書きたくない時

- CPU（Python）の複雑な作業をGPUに移植したい
- NumbaからCUDAを使ってみる
- CuPyのいろんな関数と混ぜて使いたい

- それ出来ます

[v5] Numbaとの連携

```
import cupy as cp
from numba import cuda

@cuda.jit
def square(x):
    start = cuda.grid(1)
    stride = cuda.gridsize(1)
    for i in range(start, len(x), stride):
        x[i] **= 2

a = cp.arange(5)
square[1, 32](a)
print(a)                # => [ 0  1  4  9 16]
```

CuPyのこれから

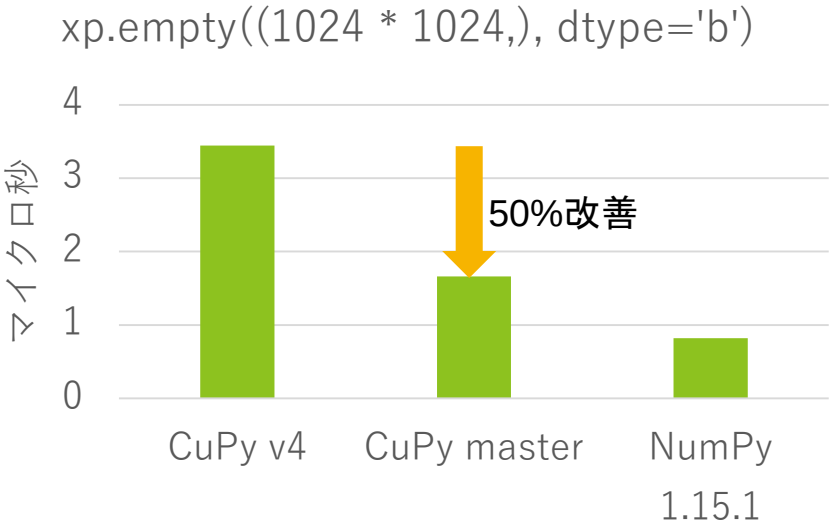
CuPy のこれから

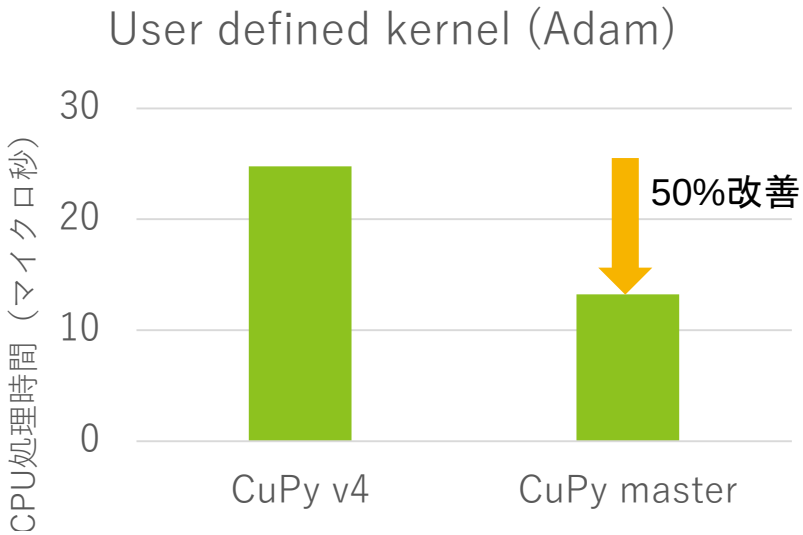
2018年10月に CuPy v5.0.0 をリリース予定

- Fusion
- Raw CUDA kernel (PyCUDAと同じ事ができます)
- 相互運用サポート
  - NumbaとのGPUデータ交換
  - DLPack：PyTorchとのデータ交換
- Windows対応
- [v5+] 関数の追加、メモリ確保、関数呼び出し速度の向上
- [v6?] 動作するGPUの種類を増やす（GTCなので小さく書いておきます）

地道な高速化

- NumPyの速度にどこまで近づけるか？





CuPyの開発者（＝私）が知りたいこと

- CuPyを何に使っているか？
- CuPyをどのように使っているか？
- CuPyに何の機能が欲しいか？
- CuPyの何を改善して欲しいか？

皆様からのフィードバックをお待ちしています

CuPyを使っている皆様にお願いしたいこと

- NvidiaやGPU関係者に「CuPyを使っています！」と言って欲しい
  - NvidiaがもっとCuPyを応援してくれるようになります

- CuPyを使ったソフトウェアを公開していたら教えて欲しい
  - CuPyが使われているソフトのリストを作っています
  - https://github.com/cupy/cupy/wiki/Projects-using-CuPy

CuPy : NumPy-like API accelerated with CUDA
(cuBLAS, cuDNN, cuRAND, cuSOLVER, cuSPARSE, cuFFT, Thrust, NCCL)

Install : $ pip install cupy
Web : https://cupy.chainer.org/
Github : https://github.com/cupy/cupy/
Example : https://github.com/cupy/cupy/tree/master/examples
Forum(ja) : https://groups.google.com/forum/#!forum/cupy-ja
Slack(ja) : https://bit.ly/chainer-jp-slack

CuPyの開発に加わりたい人歓迎です（PR・メール下さい）