Ruby Association Grant 2017年度 最終報告

Implementation of Ruby/Cumo, a CUDA-aware version of Ruby/Numo

瀬尾 直利 @sonots

プロジェクト概要(再掲)

Ruby/Numo (NUmerical MOdule) プロジェクトのうち、Future work となっている CUDA 対応版の実装を行う。 Ruby/Numo とインターフェースを合わせることにより、コードのごく一部を置き換えるだけで、GPU を使った高速化の恩恵が得られる状態を目指す。

プロジェクト計画(再掲)

Grant におけるスコープは以下の通り

- (1) CPU、GPU間のメモリ転送インターフェースの実装
- (2) GPUメモリを高速に管理するための memory pool の実装 (glibc malloc 相当の best-fit with coalescing アルゴリズムの実装)
- (3) 初期化ルーチンの実装 (empy, ones, eye, identity, zeros, fill)
- (4) 基本的な elementwise function [1] の実装
- (5) ruby コードとして記述した演算を、透過的に GPU カーネル関数として実行するための仕組みの構築

```
a = narray([1,2,3])
b = narray([1,2,3])
a + b # <= Run "+" operation in GPU
```

- (6) 基本的な broadcasting function [1] の実装
- (7) 基本的な reduction function [1] の実装。なお、reduction (sum, inner product 演算など) は GPU のような並列演算器にとっては苦手な演算となる。
- (8) cuBLAS を利用した基本的な linear algebla function の実装

以下は余力があれば実施するものとする

- (a) user defined kernel [2] のサポート. nvrtc [3] を用いてユーザが ruby 上で定義した関数を JIT コンパイル し、GPU kernel 実行する仕組みの構築
- (b) apache-arrow data frame [4] のサポート
- (c) cudnn, cusolver, cusparse, thrust, curand などその他の NVIDIA 提供ライブラリサポートの追加
- (d) kernel fusion [5]
- (e) GPU aware profiler. nvprof サポート

プロジェクト成果

ソースコード

https://github.com/sonots/cumo

プロジェクト達成状況

Grant のスコープと定義していた (1) - (8) まで全て達成。Cumo の基本的な枠組みを作れた。

追加スコープについては、(a) user-defined kernel の一部 (JIT コンパイルまで) を達成。thrust ライブラリを (7) reduction kernel の実装で利用することにしたため (c) の一部 (thrust ライブラリの利用) を達成。また、(e) nvprof サポートについては $ nvprof ruby のように nvprof 経由で ruby を起動することで特に対応することなくプロファイリング可能であったため、対応を不要とした。

CUDA Kernel 化達成度

本助成プロジェクトで作成した枠組みの上で、Numo の各関数を CUDA Kernel を利用するように書き換えていく必要がある。

現在、80ファイル中52ファイルが対応済み。

特に red-chainer [6] の mnist 実装が必要とする関数は、全て対応済みである。

Red-chainer インテグレーション

Red-chainer [6] とは Numo を利用した Ruby 製のディープラーニングフレームワークである。

Cumo のデザイン通り、red-chainer のソースコードを以下のUNIXコマンドで置換するだけで、Red-chainer を Cumo で動かすことができた。

```
find . -name '*.rb' | xargs sed -i -e  's/Numo/Cumo/g' -e 's/numo/cumo/g'
```

Numo との速度比較

以下のコードのように elementwise な掛け算でパフォーマンス評価を実施した。

```
Xumo = Numo # Cumo
Xumo::Int32.ones(size) * 2
```

Size	Numo (ms)	Cumo (ms)
10^4	0.11	0.43
10^5	0.91	0.35
10^6	8.91	1.06
10^7	80.11	7.78
10^8	721.84	74.66

CPU は Intel(R) Xeon(R) CPU E5-2623 3.00GHz

GPU は TITAN X (Maxwell)

10^8 サイズの配列において 10倍速を達成した。なお、サイズが小さい場合は Cumo の方が遅い結果になるが、これは CUDA kernel launch のオーバーヘッドが大きいためである。

プロジェクト遂行中に気づいた新たな課題

GC で GPU メモリを管理することによるメモリ再利用率の低下

Cumo では、高速化のため、GPU メモリープールを実装した。NumPy の CUDA 対応版と言える CuPy では GPU メモリープールにより速度が大幅に向上した実績があるが、Ruby は Python と異なり GC でメモリを管理するため (Python はリファレンスカウント)、GPU メモリを確保している Cumo::NArray オブジェクトが不要になっても、即座にメモリープールにメモリを返却してくれず、再利用率が CuPy に比べて低い問題があることがわかった。

そこで Cumo::NArray に free メソッドを追加し、GC に任せずにユーザが 明示的に GPU メモリを開放できるようなインターフェースを用意した。

このインターフェースは Numo::NArray にはないため、Numo と非互換となる。

メモリーコピーによる速度低下

Numo フレームワーク内部の ndloop.c でメモリのコピーが頻繁に発生しており、速度低下に影響していることがわかっている。

Numo インターフェースとの非互換

Numo には、sum を含む reduction 関数など、Numo::NArray ではなく Ruby 標準のオブジェクトを返すインターフェースがある。Ruby 標準のオブジェクトはホスト上(CPU側)のメモリを利用するため、Cumo::NArray の GPU メモリをホストに一度転送する必要があり、またCPUとGPU間の同期も取らなければならなくなるため、遅くなることがわかっている。

現状は、Numo との互換性を優先しているが、速度を優先して Cumo::NArray オブジェクトを返すオプションを用意したい。

mkmf の限界

Ruby の拡張ライブラリを開発するために、Ruby 標準の mkmf ライブラリを用いて Makefile を生成するのが一般的であるが、mkmf ライブラリに以下の課題を感じた。

1. コンパイラを 1 つしか想定しておらず、CUDA ライブラリのように gcc 以外にも nvcc も使いたい場合に素直に書けない

2. 他の Ruby 拡張ライブラリに依存した Ruby 拡張ライブラリの開発手法が明示的でない (Cumo から Numo::NArray オブジェクトを作成するために Numo を取り込むなど)

今後の計画

- Numo の関数全てを CUDA カーネル対応する
- User-defined kernel のサポート
- Numo::NArray と Cumo::NArray の変換サポート
- 高速化
    - ndloop.c におけるメモリーコピーによる速度低下を対処
    - Reduction 関数 に 0-dimensional Cumo::NArray を返すオプションの追加
    - その他細かいチューニング

謝辞

Numo の作者である田中氏とメンターである村田氏に感謝する。また、CuPy プロジェクトに感謝する。最後に、子供が小さい中、プライベートの時間を使って開発できるよう支援してくれた妻に感謝する。

参考文献

[1] http://www.scipy-lectures.org/intro/numpy/operations.html

[2] https://docs-cupy.chainer.org/en/stable/tutorial/kernel.html

[3] http://docs.nvidia.com/cuda/nvrtc/index.html

[4] https://slide.rabbit-shocker.org/authors/kou/data-science-rb/

[5] https://www.slideshare.net/pfi/automatically-fusing-functions-on-cupy

[6] https://github.com/red-data-tools/red-chainer