gpyfft Documentation

Release 0.1

Gregor Thalhammer

June 07, 2012

CONTENTS

1 gpyfft **1**

1.1 Introduction 1

1.2 Status 1

1.3 Requirements 2

1.4 Installation 2

1.5 License: 2

1.6 Tested Platforms 2

2 Building gpyfft **5**

3 gpyfft class structure **7**

3.1 GpyFFT 7

3.2 Plan 9

3.3 GpyFFT_Error 10

4 Indices and tables **11**

Bibliography **13**

Python Module Index **15**

Python Module Index **17**

CHAPTER

ONE

GPYFFT

A Python wrapper for the OpenCL FFT library APPML/clAmdFft from AMD

1.1 Introduction

AMD has created a nice FFT library for use with their OpenCL implementation called AMD Accelerated Parallel Processing Math Libraries This C library is available as precompiled binaries for Windows and Linux platforms. It is optimized for AMD GPUs. (Note: This library is not limited to work only with hardware from AMD, but according to this forum entry it currently yields wrong results on NVidia GPUs.)

This python wrapper is designed to tightly integrate with pyopencl. It consists of a low-level cython based wrapper with an interface similar to the underlying C library. On top of that it offers a high-level interface designed to work on data contained in instances of pyopencl.array.Array, a numpy work-alike array class. The high-level interface is similar to that of pyFFTW, a python wrapper for the FFTW library.

Compared to pyfft, a python implementation of Apple's FFT library, AMD's FFT library offers some additional features such as transform sizes that are powers of 2,3 and 5, and real-to-complex transforms. And on AMD hardware a better performance can be expected, e.g., gpyfft: 280 Gflops compared to pyfft: 63 GFlops (for single precision, accurate math, inplace, transform size 1024x1024, batch size 4, on AMD Cayman, HD6950).

1.2 Status

This wrapper is currently under development.

1.2.1 work done

- low level wrapper (mostly) completed
- high level wrapper: complex (single precision), interleaved data, in and out of place (some tests and benchmarking available)
- creation of pyopencl Events for synchronization

1.2.2 missing features

- debug mode to output generated kernels
- documentation for low level wrapper (instead refer to library doc)

- define API for high level interface
- high level interface: double precision data, planar data, real<->complex transforms
- high level interface: tests for non-contiguous data
- handling of batched transforms in the general case, e.g. shape (4,5,6), axes = (1,), i.e., more than one axes where no transform is performed. (not always possible with single call for arbitrary strides, need to figure out when possible)

1.3 Requirements

- python
- pyopencl (git version newer than 4 Jun 2012)
- cython
- APPML clAmdFft 1.8
- AMD APP SDK

1.4 Installation

1. Install the AMD library:
    - install clAmdFft
    - add clAmdFft/binXX to PATH, or copy clAmdFft.Runtime.dll to package directory
    - edit setup.py to point to clAmdFft and AMD APP directories

Then, either:

2. python setup.py install

Or:

3. inplace build: python setup.py build_ext –inplace

1.5 License:

LGPL

1.6 Tested Platforms

OS	Python	AMD APP	OpenCL	Device	Status
Win7 (64bit)	2.7, 64bit	2.7	OpenCL 1.2, Catalyst 12.4	AMD Cayman (6950)	works!
Win7 (64bit)	2.7, 32bit	2.7	OpenCL 1.1 AMD-APP-SDK-v2.4 (595.10)	Intel i7	works!
Win7 (64bit)	2.7, 32bit	2.7	OpenCL 1.1 (Intel)	Intel i7	works!
Win7 (64bit)	2.7, 32bit	2.7	OpenCL 1.0 CUDA 4.0.1 (NVIDIA)	Quadro 2000M	Fails
Win7 (64bit)	2.7, 32bit	2.7	OpenCL 1.2 AMD-APP (923.1)	Tahiti (7970)	works!
Win7 (64bit)	2.7, 32bit	2.7	OpenCL 1.2 AMD-APP (923.1)	AMD Phenom IIx4	works!

3. Gregor Thalhammer 2012

CHAPTER

TWO

BUILDING GPYFFT

Here will be detailed instructions for building gpyfft from source.

CHAPTER

THREE

GPYFFT CLASS STRUCTURE

3.1 GpyFFT

class `gpyfft.`**`GpyFFT`**

The GpyFFT object is the primary interface to the AMD FFT library

Methods

`get_version``()`

returns the version of the underlying AMD FFT library

Parameters **None** :

Returns **out** : tuple

the major, minor, and patch level of the AMD FFT library

Raises **GpyFFT_Error** :

An error occurred accessing the clAmdFftGetVersion function

Notes

The underlying AMD FFT call is 'clAmdFftCreateDefaultPlan'

`create_plan``()`

creates an FFT Plan object based on the requested dimensionality

Parameters **context** : pypencl.Context

http://documen.tician.de/pyopencl/runtime.html#pyopencl.Context

shape : tuple

containing from one to three integers, specifying the length of each requested dimension of the FFT

Returns **out** : gpyfft.Plan object

The generated gpyfft.Plan

Raises **None** :

3.2 Plan

class `gpyfft.`**`Plan`**

A plan is the collection of (almost) all parameters needed to specify an FFT computation. This includes:

- What pyopencl context executes the transform?
- Is this a 1D, 2D or 3D transform?
- What are the lengths or extents of the data in each dimension?
- How many datasets are being transformed?
- What is the data precision?
- Should a scaling factor be applied to the transformed data?
- Does the output transformed data replace the original input data in the same buffer (or buffers), or is the output data written to a different buffer (or buffers).
- How is the input data stored in its data buffers?
- How is the output data stored in its data buffers?

The plan does not include:

- The pyopencl handles to the input and output data buffers.
- The pyopencl handle to a temporary scratch buffer (if needed).
- Whether to execute a forward or reverse transform.

These are specified later, when the plan is executed.

Methods

`__init__`

Instantiates a Plan object

Plan objects are created internally by gpyfft; normally a user does not create these objects

Parameters **contex** : pyopencl.Context

http://documen.tician.de/pyopencl/runtime.html#pyopencl.Context

shape : tuple

the dimensionality of the transform

lib : no idea

this is a thing that does lib things

Raises **ValueError** :

when the shape isn't a tuple of length 1, 2 or 3

TypeError because the context argument isn't a valid pyopencl.Context

Notes

The underlying AMD FFT call is 'clAmdFftCreateDefaultPlan'

precision
the floating point precision of the FFT data

scale_forward
the scaling factor to be applied to the FFT data for forward transforms

scale_backward
the scaling factor to be applied to the FFT data for backward transforms

batch_size
the number of discrete arrays that this plan can handle concurrently

shape
the length of each dimension of the FFT

strides_in
the distance between consecutive elements for input buffers in a dimension

strides_out
the distance between consecutive elements for output buffers in a dimension

distances
the distance between array objects

layouts
the expected layout of the output buffers

inplace
determines if the input buffers are going to be overwritten with results (True == inplace, False == out of place)

temp_array_size
the buffer size (in bytes), which may be needed internally for an intermediate buffer

transpose_result
the final transpose setting of a multi-dimensional FFT

bake()
Prepare the plan for execution

After all plan parameters are set, the client has the option of "baking" the plan, which tells the runtime no more changes to the plan's parameters are expected, and the OpenCL kernels are to be compiled. This optional function allows the client application to perform this function when the application is being initialized instead of on the first execution. At this point, the clAmdFft runtime applies all implemented optimizations, possibly including running kernel experiments on the devices in the plan context.

Parameters **queues** : list

this is a list of things

Returns **None** :

Raises **GpyFFT_Error** :

An error occurred accessing the clAmdFftBakePlan function

Notes

The underlying AMD FFT call is 'clAmdFftBakePlan'

enqueue_transform()

Enqueue an FFT transform operation, and either return immediately, or block waiting for events.

This transform API is specific to the interleaved complex format, taking an input buffer with real and imaginary components paired together, and outputting the results into an output buffer in the same format.

Parameters **queues** : list

of things

in_buffers : array-like

array-like input data

Returns **None** :

Other Parameters **out_buffers** : array-like, optional

if the plan is out-of-place, then we have out buffers

direction_forward : bool, optional

this works like it sounds like it should

wait_for_events : list, optional

I am not sure how this interface works

temp_buffer : buffer, optional

I am not sure how this works

Raises **GpyFFT_Error** :

An error occurred accessing the clAmdFftEnqueueTransform function

Notes

The underlying AMD FFT call is 'clAmdFftEnqueueTransform'

3.3 GpyFFT_Error

class gpyfft.**GpyFFT_Error**

Exception wrapper for errors returned from underlying AMD library calls

CHAPTER

FOUR

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

g

gpyfft *(Windows)*, 7

PYTHON MODULE INDEX

g

gpyfft *(Windows)*, 7