clFFT Benchmark Script Manual

Amir Gholami
DL Library Team

July 21, 2015

Contents

1 Introduction

The python scripts are intended to automate benchmarking of clFFT and cuFFT libraries for a range of configurations. These include 1/2/3D, real-to-complex, complex-to-complex, and single/double transforms. Moreover, the scripts include an automated plotting tool for comparing clFFT and cuFFT. It is assumed that the user has already installed the libraries and built the corresponding clients.
The package includes the following python scripts:

- measurePerformance.py (main file)
- fftPerformanceTesting.py
- fftPerformanceTesting.pyc
- performanceUtility.pyc
- performanceUtility.py
- plotPerformance.py
- errorHandler.py
- errorHandler.pyc
- launch.sh (optional)
- fig_launch.sh (optional)

A detailed explanation of how to use these scripts is given in the next section.

2 Measure Performance Manual

The following arguments can be passed to *measureperforamcne.py*:

- **-h, –help**
  show this help message and exit
- **–device DEVICE**
  device(s) to run on; may be a comma-delimited list. choices are ['g', 'c']. (default gpu)
- **-b BATCHSIZE, –batchsize BATCHSIZE**
  number of FFTs to perform with one invocation of the client. the special value 'adapt' may be used to adjust the batch size on a per-transform basis to the maximum problem size possible on the device. To have an adaptive batch size different than the max, you can use the adaptivemax option. (default 1)
- **-a CONSTPROBSIZE, –adaptivemax CONSTPROBSIZE**
  Max problem size that you want to maintain across the invocations of client with different lengths. This is adaptive and adjusts the batch automtically such that $X \times Y \times Z \times b = a$.(default -1, i.e. do not change batch)
- **-x LENGTHX, –lengthx LENGTHX**
  length(s) of x to test; must be factors of 1, 2, 3, or 5 with clFFT; may be a range or a comma-delimited list. e.g., 16-128 or 1200 or 16,2048-32768 (default 1)
- **-y LENGTHY, –lengthy LENGTHY**
  length(s) of y to test; must be factors of 1, 2, 3, or 5 with clFFT; may be a range or a comma-delimited list. e.g., 16-128 or 1200 or 16,32768 (default 1)
- **-z LENGTHZ, –lengthz LENGTHZ**
  length(s) of z to test; must be factors of 1, 2, 3, or 5 with clFFT; may be a range or a comma-delimited list. e.g., 16-128 or 1200 or 16,32768 (default 1)
- **-reps REPS**
  Number of repetitions of kernel execution (default 10)
- **-prime_factor PRIME_FACTOR, –prime_factor PRIME_FACTOR**

  only test the prime factors within the specified range of lengthx/y/z. Select from 2,3,5, and 7. Example: -prime_factor 2,3
- **-test_count TEST_COUNT, –test_count TEST_COUNT**
  Maximum number of tests to perform

- **-i INPUTLAYOUT, –inputlayout INPUTLAYOUT**
  Select from the following:
  1. interleaved (default)
  2. planar
  3. hermitian interleaved
  4. hermitian planar
  5. real
- **-o OUTPUTLAYOUT, –outputlayout OUTPUTLAYOUT**
  Select from the following:
  1. interleaved (default)
  2. planar
  3. hermitian interleaved
  4. hermitian planar
  5. real
- **–placeness PLACENESS**
  Choices are ['in', 'out']. in = in place, out = out of place (default in)
- **-r PRECISION, –precision PRECISION**
  Choices are ['single', 'double']. (default single)
- **–library clFFT,cuFFT**
  indicates the library to use for testing on this run
- **–label LABEL**
  a label to be associated with all transforms performed in this run. if LABEL includes any spaces, it must be in "double quotes". note that the label is not saved to an .ini file. e.g., –label cayman may indicate that a test was performed on a cayman card or –label "Windows 32" may indicate that the test was performed on Windows 32
- **–ini INIFILENAME**
  use the parameters in the named .ini file instead of the command line parameters.
- **–tablefile TABLEOUTPUTFILENAME**
  save the results to a plaintext table with the file name indicated. this can be used with plotPerformance.py to generate graphs of the data (default: table prints to screen)
- **–prefix PREFIX**
  Path where the library client is located (default current directory)

2.1 Example Uses

- **1D Complex-to-Complex transform, for powers of 2 in the range of 2-64 with an adaptive batch size:**
  python measurePerformance.py -x 2-64 -b 10 –library clFFT -prime_factor 2
- **2D Complex-to-Complex transform, for powers of 3 in the range of 2-8192 with an adaptive batch size:**
  python measurePerformance.py -x 2-8192 -y 2-8192 -b adapt –library clFFT -prime_factor 3
- **3D Real-to-Complex transform, for powers of 2 in the range of 2-8192 with an adaptive batch size:**
  python measurePerformance.py -x 2-8192 -y 2-8192 -z 2-8192 -b adapt –library clFFT -prime_factor 3 -i 5 -o 3
- **Read the test cases from a table file:**
  python measurePerformance.py –ini input.csv

lengthx	lengthy	lengthz	batch	device	inlay	outlay	place	precision	gflops
2	1	1	10	g	1	1	in	single	74.8
4	1	1	10	g	1	1	in	single	150.8
8	1	1	10	g	1	1	in	single	191.2
16	1	1	10	g	1	1	in	single	268.5
32	1	1	10	g	1	1	in	single	305.8
64	1	1	10	g	1	1	in	single	489.4

Table 1: Sample output generated by *measureperformance.py* script by running the first command.

lengthx	lengthy	lengthz	batch	device	inlay	outlay	place	precision	label
2	1	1	1024	g	1	1	in	single	test
4	1	1	512	g	1	1	in	single	test
8	1	1	256	g	1	1	in	single	test
16	8	1	64	g	1	1	in	single	test
16	8	4	32	g	1	1	in	single	test

Table 2: Sample input csv file that can be used as input to *measureperformance.py*, to test specific configurations.

An example output of *measureperformance.py* script is shown in Table 1. Note that for the last example the input file (input.csv) must be a comma separated csv file, including all the input specifications. An exapmle is shown in Table 2.

3 Graph Generation

The *plotperformance.py* script, is written for generating graphs directly from the csv output of *measureperformance.py*. The script has the following options:

- **-h, –help**
  show help message and exit
- **-d DATAFILE, –datafile DATAFILE**
  indicate a file to use as input. must be in the format output by measurePerformance.py. may be used multiple times to indicate multiple input files. e.g., -d clFFTOutput.txt -d cuFFTOutput.txt
- **-x x,y,z,batchsize,problemsize, –x_axis x,y,z,batchsize,problemsize**
  indicate which value will be represented on the x axis. problemsize is defined as x*y*z*batchsize
- **-y gflops, –y_axis gflops**
  indicate which value will be represented on the y axis
- **–plot device,precision,label**
  indicate which of ['device', 'precision', 'label'] should be used to differentiate multiple plots. this will be chosen automatically if not specified
- **–title GRAPHTITLE the desired title for the graph generated by this**
  execution. if GRAPHTITLE contains any spaces, it must be entered in "double quotes". if this option is not specified, the title will be auto-generated
- **–x_axis_label XAXISLABEL**
  the desired label for the graph's x-axis. if XAXISLABEL contains any spaces, it must be entered in "double quotes". if this option is not specified, the x-axis label will be auto-generated
- **–x_axis_scale linear,log2,log10**
  the desired scale for the graph's x-axis. if nothing is specified, it will be selected automatically
- **–y_axis_scale linear,log2,log10**
  the desired scale for the graph's y-axis. if nothing is specified, linear will be selected

- **–y_axis_label YAXISLABEL**
  the desired label for the graph's y-axis. if YAXISLABEL contains any spaces, it must be entered in "double quotes". if this option is not specified, the y-axis label will be auto-generated

- **–outputfile OUTPUTFILENAME**
  name of the file to output graphs. Supported formats: emf, eps, pdf, png, ps, raw, rgba, svg, svgz.

3.1 Example Uses

- **Plot Gflops for different sizes:**
  python plotperformance.py -d clFFT_performance.txt -x x -y gflops –x_axis_scale log2 –y_axis_scale linear

- **Plot Gflops for different sizes and write to a file:**
  python plotperformance.py -d clFFT_performance.txt -x x -y gflops –x_axis_scale log2 –y_axis_scale linear –outputfile "hello_world.png"

- **Comparison plot:**
  python plotperformance.py -d clFFT_performance.txt -d cuFFT_performance -x x -y gflops –x_axis_scale log2 –y_axis_scale linear –outputfile "hello_world.png"

An example graph for the last case is shown in Figure 1. Two optional shell scripts are shipped with the package named *launch.sh* and *fig_launch.sh*. These scripts perform the benchmarking of clFFT and cuFFT for 1/2/3D, R2C, C2C transforms and put the results into clFFT_benchmark and cuFFT_benchmark folders (in the same path). Executing *fig_launch.sh* would create all the 12 comparison graphs between the two libraries.

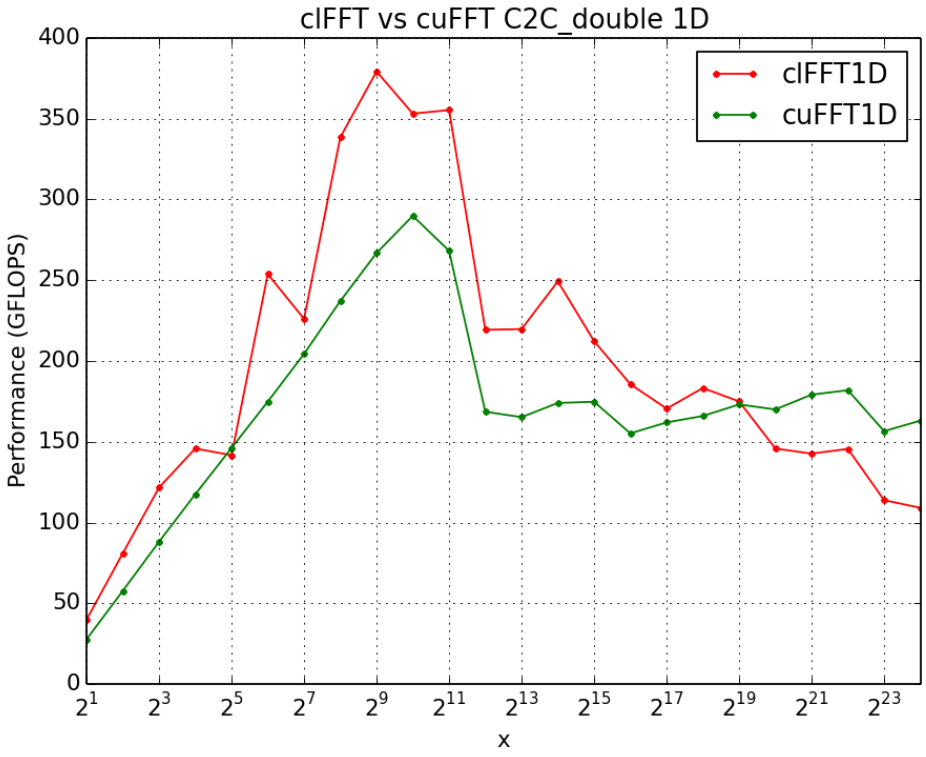

Figure 1: Sample figure generated by *plotperformance.py* comparing clFFT and cuFFT.