clMathLibraries / clBLAS

Watch 84 Star 518 Fork 159

Code Issues 63 Pull requests 0 Projects 0 Wiki Pulse Graphs

a software library containing BLAS functions written in OpenCL

371 commits **4** branches **7** releases **31** contributors Apache-2.0

Branch: master New pull request Create new file Upload files Find file Clone or download

tingxingdong committed on **GitHub** Merge pull request #312 from clMathLibraries/revert-311-device-all ... Latest commit cf91139 on 24 Mar

doc	add perf data for fiji	a year ago
src	Revert "Use CL_DEVICE_TYPE_ALL for all samples"	2 months ago
.gitattributes	Initial check-in of open source clBLAS code	4 years ago
.gitignore	Removing the pedantic flag from gcc compiles	a year ago
.travis.yml	Update .travis.yml and appveyor.yml	a year ago
CHANGELOG	Cleanup of txt not in the spirit of the clMath Apache license	4 years ago
CONTRIBUTING.md	Point the CONTRIBUTING wiki links to the correct repository	9 months ago
LICENSE	delete appendix in license file	2 years ago
NOTICE	Initial check-in of open source clBLAS code	4 years ago
README.md	Update README with release notes	4 months ago
appveyor.yml	Update .travis.yml and appveyor.yml	a year ago

README.md

Build Status

Build branch	master	develop
GCC/Clang x64	build passing	build passing
Visual Studio x64	build passing	build passing

clBLAS

This repository houses the code for the OpenCL™ BLAS portion of clMath. The complete set of BLAS level 1, 2 & 3 routines is implemented. Please see Netlib BLAS for the list of supported routines. In addition to GPU devices, the library also supports running on CPU devices to facilitate debugging and multicore programming. APPML 1.12 is the most current generally available pre-packaged binary version of the library available for download for both Linux and Windows platforms.

The primary goal of clBLAS is to make it easier for developers to utilize the inherent performance and power efficiency benefits of heterogeneous computing. clBLAS interfaces do not hide nor wrap OpenCL interfaces, but rather leaves OpenCL state management to the control of the user to allow for maximum performance and flexibility. The clBLAS library does generate and enqueue optimized OpenCL kernels, relieving the user from the task of writing, optimizing and maintaining kernel code themselves.

clBLAS update notes 01/2017

- v2.12 is a bugfix release as a rollup of all fixes in /develop branch
  - Thanks to @pavanky, @iotamudelta, @shahsan10, @psyhtest, @haahh, @hughperkins, @tfauck @abhiShandy, @IvanVergiliev, @zougloub, @mgates3 for contributions to clBLAS v2.12
- Summary of fixes available to read on the releases tab

clBLAS library user documentation

Library and API documentation for developers is available online as a GitHub Pages website

Google Groups

Two mailing lists have been created for the clMath projects:

- clmath@googlegroups.com - group whose focus is to answer questions on using the library or reporting issues
- clmath-developers@googlegroups.com - group whose focus is for developers interested in contributing to the library code itself

clBLAS Wiki

The project wiki contains helpful documentation, including a build primer

Contributing code

Please refer to and read the Contributing document for guidelines on how to contribute code to this open source project. The code in the /master branch is considered to be stable, and all pull-requests should be made against the /develop branch.

License

The source for clBLAS is licensed under the Apache License, Version 2.0

Example

The simple example below shows how to use clBLAS to compute an OpenCL accelerated SGEMM

```
#include <sys/types.h>
#include <stdio.h>

/* Include the clBLAS header. It includes the appropriate OpenCL headers */
#include <clBLAS.h>

/* This example uses predefined matrices and their characteristics for
 * simplicity purpose.
*/

#define M  4
#define N  3
#define K  5

static const cl_float alpha = 10;

static const cl_float A[M*K] = {
11, 12, 13, 14, 15,
21, 22, 23, 24, 25,
31, 32, 33, 34, 35,
41, 42, 43, 44, 45,
};
static const size_t lda = K;        /* i.e. lda = K */

static const cl_float B[K*N] = {
11, 12, 13,
21, 22, 23,
31, 32, 33,
41, 42, 43,
51, 52, 53,
};
static const size_t ldb = N;        /* i.e. ldb = N */

static const cl_float beta = 20;

static cl_float C[M*N] = {
    11, 12, 13,
```

```
    21, 22, 23,
    31, 32, 33,
    41, 42, 43,
};
static const size_t ldc = N;        /* i.e. ldc = N */

static cl_float result[M*N];

int main( void )
{
cl_int err;
cl_platform_id platform = 0;
cl_device_id device = 0;
cl_context_properties props[3] = { CL_CONTEXT_PLATFORM, 0, 0 };
cl_context ctx = 0;
cl_command_queue queue = 0;
cl_mem bufA, bufB, bufC;
cl_event event = NULL;
int ret = 0;

/* Setup OpenCL environment. */
err = clGetPlatformIDs( 1, &platform, NULL );
err = clGetDeviceIDs( platform, CL_DEVICE_TYPE_GPU, 1, &device, NULL );

props[1] = (cl_context_properties)platform;
ctx = clCreateContext( props, 1, &device, NULL, NULL, &err );
queue = clCreateCommandQueue( ctx, device, 0, &err );

/* Setup clBLAS */
err = clblasSetup( );

/* Prepare OpenCL memory objects and place matrices inside them. */
bufA = clCreateBuffer( ctx, CL_MEM_READ_ONLY, M * K * sizeof(*A),
                      NULL, &err );
bufB = clCreateBuffer( ctx, CL_MEM_READ_ONLY, K * N * sizeof(*B),
                      NULL, &err );
bufC = clCreateBuffer( ctx, CL_MEM_READ_WRITE, M * N * sizeof(*C),
                      NULL, &err );

err = clEnqueueWriteBuffer( queue, bufA, CL_TRUE, 0,
    M * K * sizeof( *A ), A, 0, NULL, NULL );
err = clEnqueueWriteBuffer( queue, bufB, CL_TRUE, 0,
    K * N * sizeof( *B ), B, 0, NULL, NULL );
err = clEnqueueWriteBuffer( queue, bufC, CL_TRUE, 0,
    M * N * sizeof( *C ), C, 0, NULL, NULL );

    /* Call clBLAS extended function. Perform gemm for the lower right sub-matrices */
    err = clblasSgemm( clblasRowMajor, clblasNoTrans, clblasNoTrans,
                           M, N, K,
                           alpha, bufA, 0, lda,
                           bufB, 0, ldb, beta,
                           bufC, 0, ldc,
                           1, &queue, 0, NULL, &event );

/* Wait for calculations to be finished. */
err = clWaitForEvents( 1, &event );

/* Fetch results of calculations from GPU memory. */
err = clEnqueueReadBuffer( queue, bufC, CL_TRUE, 0,
                            M * N * sizeof(*result),
                            result, 0, NULL, NULL );

/* Release OpenCL memory objects. */
clReleaseMemObject( bufC );
clReleaseMemObject( bufB );
clReleaseMemObject( bufA );

/* Finalize work with clBLAS */
clblasTeardown( );

/* Release OpenCL working objects. */
clReleaseCommandQueue( queue );
clReleaseContext( ctx );

return ret;
```

```
    }
```

Build dependencies

Library for Windows

- Windows® 7/8
- Visual Studio 2010 SP1, 2012
- An OpenCL SDK, such as APP SDK 2.8
- Latest CMake

Library for Linux

- GCC 4.6 and onwards
- An OpenCL SDK, such as APP SDK 2.9
- Latest CMake

Library for Mac OSX

- Recommended to generate Unix makefiles with cmake

Test infrastructure

- Googletest v1.6
- Latest Boost
- CPU BLAS
- Netlib CBLAS (recommended) Ubuntu: install by "apt-get install libblas-dev" Windows: download & install lapack-3.6.0 which comes with CBLAS
- or ACML on windows/linux; Accelerate on Mac OSX

Performance infrastructure

- Python