**Accelerated IDL using OpenCL**

Michael D. Galloy, mgalloy@txcorp.com
Tech-X Corporation

Work supported by NASA SBIR Grant #NNX12CA79C
IN53B-1563

---

**Introduction**

The Interactive Data Language (IDL) is also widely used in this community for data analysis and visualization. Previously, we have developed GPULib, a library of routines in IDL for accelerating common scientific operations including arithmetic, FFTs, interpolation, and other computations using CUDA-enabled GPUs from NVIDIA.

We have now ported portions of this library to OpenCL, making GPU accelerated IDL available on any modern GPU or CPU. Currently, our OpenCL library supports basic array operations and the ability to execute user-defined OpenCL kernels.

---

**CUDA and IDL**

IDL is a high-level array-based language used predominantly by researchers for scientific/image-processing calculations and visualizations. NVIDIA's CUDA parallel computing platform and programming model enables large increases in computing performance by using NVIDIA's graphics processing units (GPUs).

**CUDAlib and IDL**

IDL is an open-source library designed to enable researchers to quickly write software in the interactive and familiar environment of IDL, while using the computing power of modern GPUs.

**OpenCL**

OpenCL is an open standard and runs on multiple vendor hardware implementations including both CPU and GPU hardware, and is available for both CPUs and GPUs.

Use of the open standard OpenCL implemented by several hardware vendors from within GPULib is an exciting prospect with several advantages: 1) OpenCL is an open standard and runs on multiple vendor hardware implementations including both CPU and GPU hardware, 2) OpenCL uses runtime compilation of kernels allowing easy construction of custom kernels at runtime (useful for creating user-defined filtering functions) In addition, we have a library of GPU accelerated routines (using CUDA) to perform simple operations with OpenCL.

---

**Roadmap**

Future features of GPULib planned:

- Support for integer types (all numeric types of IDL)
- Leveren-Marquardt curve fitting
- CUDA, MAGMA, and IDL updates as needed
- Ease of integration with mpiDLTaskDL to support multiple GPUs
- OpenCL support

---

**GPULib**

**Current capabilities**

The features of GPULib include:

- Accelerated basic vector, as well as matrix, arithmetic for float, double, complex, and double complex arrays
- Ability to run user-defined CUDA kernels
- Accelerated FFT (1-, 2-, and 3-dimensional, as well as batched)
- LAPACK routines (MAGMA bindings)
- Increased memory efficiency across all routines (useful for creating user-defined filtering functions)
- Reconstruction of 2-dimension object using a GPU accelerated Radon transform

---

**GPU computing in IDL**

Modern commodity hardware devices, GPUs and CPUs—standalone or combined into clusters, offer unprecedented computing power. Performance has grown exponentially, as seen in the graph below. But parallel programming frameworks are required to harness the parallel nature of the new hardware, as single core performance has been stagnant for the last decade.

---

**TaskDL**

TaskDL is a task farming library for homogeneously parallel (“embarrassingly parallel”) problems which require no communication between workers. These loosely-coupled applications can run on nodes of a cluster or cores of a single machine.

Available for Windows, Linux, or OS X.

**mpiDL**

mpiDL was developed to implement parallel algorithms whose calculations require communication between processors. It leverages the power of the industry-standard MPICH-Passing Interface (MPI) with the ease of use and variability present in MPI.

Available for Linux or OS X.

**Combining technologies**

Either TaskDL or mpiDL can also be used in combination with GPULib to control multiple GPUs. The OpenCL prototype allows for even more utilization of devices in a heterogeneous computing environment—in this environment the same code can be used on all devices.

---

**References**

- [LAPACK routines provided by MAGMA and simple ways to access user-written](http://example.com)
- [In addition, we have a library of GPU accelerated routines (using CUDA)](http://example.com)
- [The disadvantages of using OpenCL are also significant, but are being improved rapidly:](http://example.com)
- [Michael D. Galloy, mgalloy@txcorp.com](mailto:mgalloy@txcorp.com)
- [Tech-X Corporation](http://www.tech-x.com)
- [Copyright © 2013 Tech-X Corporation](http://www.tech-x.com)

---

**Figures and Tables**

- [Graph of CUDA vs. CPU performance for SIMD, vectorized and unoptimized CPU](http://example.com)
- [Graph showing performance of IDL bindings for various operations](http://example.com)

---

**Code Examples**

```idl
IDL> y = findgen(10)
IDL> x = findgen(10)
```

```idl
dz = dx + dy
```

```idl
dx = cl_findgen(10)
```

---

**Performance**

- **CUDA**
- **OpenCL**
- **CPU**

---

**Graphs**

- [Graph showing performance of CUDA vs. OpenCL vs. CPU](http://example.com)
- [Graph comparing performance of OpenCL vs. CUDA vs. CPU for various operations](http://example.com)