A Short Introduction to the *gpuR* Package

Dr. Charles Determan Jr. PhD*

February 9, 2016

1 Introduction

GPUs (Graphic Processing Units) were originally developed to perform graphic rendering and commonly referred to in the comuting gaming world. These devices are also able to be applied to numerical operations in parallel. Although there are a few different vendors, the two primary competitors are AMD and NVIDIA.

NVIDIA GPUs depend upon the proprietary CUDA framework whereas AMD GPUs utilize the open source OpenCL framework. The upside of OpenCL is that 'kernels' are able to used on any GPU whereas CUDA kernels can only be used on NVIDIA GPUs. It is worth noting, however, that CUDA tends to edge out OpenCL performance likely a result of the highly specific framework to the NVIDIA GPUs.

That said, programming either framework is often difficult for programmers unaccustomed to working with such a low-level interface. Creating bindings for high-level programming languages (such as R) make using GPUs much more accessible to a broader audience.

Several R packages have been developed including *gputools*, *cudaBayesreg*, *HiPLARM*, *HiPLARb*, and *gmatrix*. However, all of the packages depend upon a CUDA backend and therefore restrict the using to only using NVIDIA GPUs. The novelty of this package is the use of the ViennaCL library (http://viennacl.sourceforge.net/) which has been coveniently been repackaged in the *RViennaCL* package to be used in other R packages. This allows the user to leverage the auto-tuned OpenCL kernels of the ViennaCL library on any GPU. It also allows for a CUDA backend for those who in fact have a NVIDIA GPU for improved performance, which will be provided in a companion *gpuRcuda* package.

Of the aforementioned packages, most contain a very limited set of functions available to the R user within the packages. The most extensive being the *gmatrix* package which contains most linear algebra operations. All of the packages (with the exception of *gmatrix*) don't store the data on the GPU. As such, there is the overhead of transferring data back and forth between the device and host. Similar to *gmatrix*, this package utilizes S4 classes to store an external pointer to the data on the GPU which mirror the base `matrix` and `vector` classes. However, given the interactive nature of R programming and

*cdetermanjr@gmail.com

the limited RAM available on GPU's this package provides intermediate classes that remove the object from GPU RAM to allow objects to be stored on the CPU but still utilize the GPU as needed.

2 Install

Install *gpuR* using

```
# Stable version
install.packages("gpuR")

# Dev version
devtools::install_github("cdeterman/gpuR")
```

3 Basic Use with *gpuMatrix*

gpuR has most basic linear algebra operations. The user simply needs to create a *gpuMatrix* object and the GPU methods will be used. Here is a minimal example demonstrating typical matrix multiplication.

```
library("gpuR")

# verify you have valid GPUs
detectGPUs()

# create gpuMatrix and multiply
set.seed(123)
gpuA <- gpuMatrix(rnorm(16), nrow=4, ncol=4)
gpuB <- gpuA %*% gpuA
```

Most linear algebra methods have been created to be executed for the *gpuMatrix* and *gpuVector* objects. These methods include basic Aithmetic functions `%*%`, `+`, `-`, `*`, `/`, `crossprod`, `tcrossprod`, `colMeans`, `colSums`, `rowMean`, and `rowSums`. Math functions include `sin`, `asin`, `sinh`, `cos`, `acos`, `cosh`, `tan`, `atan`, `tanh`, `exp`, `log`, `log10`. Additional operations include some linear algebra routines such as `cov` (Pearson Covariance) and `eigen`. With many more in development.

The objects may also be created specifying data type including int, float, and double. Float type was included to provide a smaller memory footprint and also increased throughput if the increased accuracy of double is not required.

Both the *gpuMatrix* and *vclMatrix* objects return a pointer to the data. Given that working with GPU's implies that you are working with larger datasets, this prevents R from making unneccessary copies. However, this does require the user to exercise caution as any change made to a 'copy' (e.g. `gpuB <- gpuA`) will result in changes to the original object and all others pointing to it as well.

4 *vclMatrix* Class

The *vclMatrix* class was created to allow the user to put data directly on the GPU once and not need to continually push data back and forth between the host and device. Therefore, if multiple processes are to be applied to a given matrix, there will be significant savings by using *vclMatrix* objects. It is important to remember though, different GPU's have different amounts of RAM. The interactive nature of R often has many objects existing simultaenously where you may exceed your GPU's RAM. As such, the *gpuMatrix* class is provided.