R2GPU: A Very Simple R Interface for General Purpose Computing on Nvidia GPUs
Julio Olaya, Xueyuan Cao, Stan Pounds
Department of Biostatistics, St. Jude Children’s Research Hospital, Memphis, TN, USA

Abstract
We have developed the R2GPU suite of R packages (R2GPUbase, R2GPUmath, R2GPUprob) to provide a very simple R interface to general purpose computing on NVIDIA graphical processing units (GPUs). The R2GPU packages define the infrastructure to initiate interactions between an R session and a GPU device, transmit data between an R session and a GPU device, and transmit instructions from R to a GPU device. This infrastructure provides two distinct practical advantages to the R programmer:
(1) most functions for the GPU are syntactically identical to standard R, thereby reducing the burden of learning a new set of R commands; and
(2) the results of a GPU calculation may remain on the GPU for further processing by the GPU, thereby greatly reducing the overhead of using the GPU to maximize the speed-up for real applications.

In this way, the R2GPU packages provide R programmers the ability to easily expand existing packages to perform computationally intensive tasks on the GPU.

Objectives
1. Use standard R syntax for GPU computing.
2. Maximize speed-up for real applications.

Methods
1. Define an R object class CP that points from the CPU to data on the GPU.
2. Define toGPU (x) as a function that transmits the data in x from the CPU to the GPU and returns a CP object.
3. Use a standard R interface to define methods for CP objects.
4. Define toGPU (x) as a function that operates on the CP object class to bring data back to the CPU from the GPU.

Illustrative Example
# Rank and z-transform each row or column of a matrix
def prep.mtx=preprocess(x, de, w) { # a numeric matrix
def w=apply(x, 1, rank) # find mean of ranks for each row
z=ran(x) # rank each row on GPU
def t(z) { # transpose
dm=(n*m) # number of columns
z=runif(dm, 0, 1) # generate uniform random numbers
z=ran(x) # rank each row on GPU
z=z==2 # initialize z
if (!z) # move data to GPU
z=t(z) # transpose to operate on columns
else # find mean of ranks for each row
if (z) # rank each row on GPU
z=t(z) # transpose back if
else # find mean of ranks for each row
z=t(z) # transpose to operate on columns

m=ran(dm) # random matrix
s=ran(m) # mean of ranks for each row
s=s==ran(m) # rank each row on GPU
z=ran(z) # rank each row on GPU
z=t(z) # transpose back if
else # find mean of ranks for each row
z=t(z) # transpose back if
return(z) # return
}

GPU: 2.14
CPU: 22.13
m = 100,000 rows; n = 1,000 columns

Conclusions and Discussion
1. The R2GPU packages empower R programmers to achieve massive speed-ups (10-100 fold) on a GPU while leaving most of their CPU code unchanged.
2. GPUs are much more affordable parallel computing resource than traditional clusters. A K40 GPU costs roughly $3,500.
3. Future work will make the GPU available for more standard R functions.

Example Application
We used the R2GPU packages to develop a GPU implementation of the projection onto the most interesting statistical evidence (PROMISE; Pounds et al. 2009; PMID 19528606) statistical method. PROMISE is an integrated data analysis method that identifies genes that show the most significant pattern of biologically meaningful associations with multiple pharmaceutical and clinical endpoint variables. PROMISE is a computationally intense method based on permutation testing. We developed an implementation that could use the CPU or GPU. The GPU was 125 times faster than the CPU. The vast majority of the code lines were identical for the GPU and CPU implementations (Figure 3).

Technical Details
Hardware: Dell cluster with two nodes. Each node has 12 CPU cores (2.4 GHz Intel Xeon). Eight NVIDIA K40m Tesla GPUs. Each K40m has 11.34 GB Global Memory.
Software: R Linux Version 3.1.2 CUDA v6.5 runtime APIs, computer capabilities of 3.5.

Acknowledgements: St. Jude donors, ALSAC, the PBTC Foundation, and Dr. James Boyett