GPU computing for R Statistical Environment

Jaideep Singh*, Ipseeta Aruni
Department of Electrical Engineering
Indian Institute of Technology, Roorkee
HPC-Project, Montpellier, France
* Contact author: jaiitug@gmail.com

Keywords: GPU, SWIG, CUDA, CUBLAS, **R.oo**

The idea of offering Graphics Processing Unit (GPU) computing solutions using CUDA to *R* users has been in the form of adding new functions to *R* that can be used to transfer data and perform computations on the GPUs (http://brainarray.mbni.med.umich.edu/brainarray/rgpgpu/). A better solution could be to introduce a new datatype in *R* that behaves the same way as *R* default datatypes. Also, it should enable *R* users to perform basic operations like addition, multiplication, sub-referencing and others using the same operators as for *R* default datatypes. **We can use the same function names for the new datatype and perform computation on the GPUs as the call to these functions would invoke the appropriate function based on the datatype in the parameter list**. Under the hood, we could write new versions of these functions to manipulate data and perform operations on the GPUs.

With this idea in mind, we extended *R* using **R.oo** and Simplified Wrapper and Interface Generator (SWIG). **R.oo** package allowed us to introduce a new class "GPU" and overload the basic *R* operators for the new datatype. SWIG was used to make C++ classes and add various functions to allocate space in GPU memory and perform computations on the GPUs.

"GPU" class has a data member that holds an external pointer. This pointer points to objects created using C++ classes and returned as an external pointer through the SWIG interface. A schematic diagram of the whole design is depicted in Fig. 1. The SWIG interface was modified to allow us to return data as *R* default datatypes as well as an external pointer. The fact that *R* has garbage collector made the integration complex, but was dealt with by adding a member flag to the C++ class Box that is set/reset during memory allocations and used to call `gc()` if required.

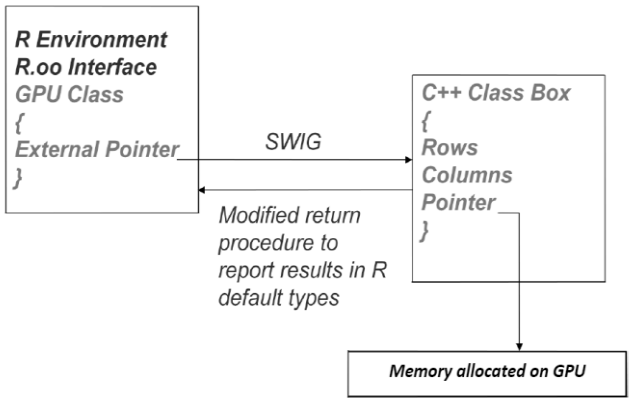

Fig 1. Schematic diagram depicting the interaction between R and C++ class using SWIG and **R.oo**

```
Example:  R>
              # To make a GPU matrix
                a=GPU(3,3);
R> a[,]
                a =
                Rows=3, Columns=3
                1 0 0
                0 1 0
                0 0 1
R> a[1,]
                a =
                Rows=1, Columns=3
                1  0  0
R> a[1:2,1:2]
                a =
                Rows=2, Columns=2
                1  0
                0  1
```

Fig 2. Sub-reference operations on GPU matrix

An example to prove the usefulness and capabilities of this design is shown in Fig. 2. Here we perform sub-referencing operation on the new datatype using overloaded operator '['. To get back the results computed on the GPUs in the form of *R* default datatypes, the *R* users can use `gresult(GPU object)` as shown below. We have designed the functions for performing the common arithmetic operations that operate on the new datatype, computing results on the GPUs and also integrated CUBLAS functions achieving impressive performance gains. Most commonly used *R* functions for which the GPU versions have already been developed or can be developed; can be easily integrated into this package. This work was done during our internship at HPC-Project, Montpellier, France, under the guidance of Mr. Thierry Porcher, Chief Technical Officer, HPC-Project.

"The R user doesn't need to bother about function names and the datatypes supplied as parameters"
"He can now move the data and computations onto the GPUs effortlessly"

```
Example:
R> a = gresult(GPU(2,2))
R> a
     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

References

R.oo: R object-oriented programming with or without references,
http://cran.r-project.org/web/packages/R.oo/index.html
Welcome to SWIG,
http://www.swig.org/