R package 'rgpu' release notes

M Kempenaar*, M Dijkstra
*contact: m.kempenaar@med.umcg.nl

February 23, 2010

Contents

1 R/GPU version 0.8-1 **2**
1.1 Features and changes 2
1.2 Outstanding issues 4
1.3 System requirements - Windows 5
1.4 System requirements - Linux / Mac OSX 5
1.5 Installation Guide - Windows 5
1.6 Installation Guide - Linux / Mac OSX 6

2 R/GPU version 0.1-2 **7**
2.1 Features 7
2.2 System requirements 7
2.3 Installation guide - Linux / Mac OSX 8

1 R/GPU version 0.8-1

This release of R/GPU features many updates and bug fixes as well as new functionality. The most major changes are the fully updated `evalgpu` function and support for the Windows operating system. Section 1.1 describes the changes in this release in comparison to version 0.1.2 (see Section 2).

1.1 Features and changes

This section describes recent changes to the R/GPU package that changes its usability or functionality. Other than the changes listed below, numerous changes have been made since the previous release to remove bugs and increase general performance.

Windows support This release of R/GPU offers support for the Windows operating systems. Current version support Windows XP, Vista and the recently released Windows 7. This binary release is compiled for 32 bits, but supports 64 bits versions as well.

evalgpu This major function of the R/GPU package has been completely redesigned. The new version offers better usability and extended functionality. Currently, `evalgpu` supports any R expression as input containing not only operators but also functions. Although `evalgpu` supports all R functions, functions that cannot be evaluated on the GPU will be evaluated using the CPU and R. The complete expression is recursively parsed which for example allows arguments to non-GPU functions to be evaluated on the GPU. For example in the expression `qnorm(sqrt(x))`, the `sqrt(x)` part will be evaluated on the GPU after which the result is passed to the `qnorm` function for evaluation on the CPU. This behavior works with all R functions.

`evalgpu` also simplifies expressions to optimize the performance. The function will recognize any constants in the expression such as part of the density normal distribution expression: $1/(\sigma\sqrt{2\pi})$. Such a constant will be evaluated only once on the CPU instead of for each element on the GPU. In some expressions this reduces the total number of entities by 50%.

`evalgpu` can also be used inside other functions which allows existing functions to be easily modified to run on the GPU. See for example the included `dnormgpu` function which is simply a wrapper for the `evalgpu` function to calculate the density normal distribution on the GPU.

gemmgpu Replacing the `mmult_g` function, `gemmgpu` offers *General Matrix Multiplication* (GEMM). The GEMM routine calculates the new value of matrix C based on the matrix-product of matrices A and B and the old value of matrix C:

$$C \leftarrow \alpha * op(A) * op(B) + \beta C$$

where $op(X) = X$ or $op(X) = X^T$ and α and β are scalar coefficients. The function arguments `transa` and `transb` determines if the matrices A and B are to be transposed. The `gemmgpu` function can easily be used for basic matrix multiplication by only using matrices A and B as input.

Restrictions to the dimensions of the input matrices are as follows: the number of columns of matrix B and matrix C should be equal and must be at least zero, also, the number of columns of matrix A and the number of rows of matrix B should be equal and must be at least zero.

This function is available in single precision (SGEMM) and double precision (DGEMM) depending on the installation of the package. Complex numerical data is unsupported at this time. See the manual page in R with `?gemmgpu` for a full description.

exportgpu This function replaces the `cuda_export` function. Arguments can now be entered without quotes. Also, exporting an already exported data object will first remove the existing copy and replace it with a new one.

lsgpu The function `lsgpu` replaces the `cuda_view` function and now returns a character list instead of printing details on data objects resident in GPU memory. This list can be used as an argument for `rmgpu` to remove all objects at once.

rmgpu Arguments to `rmgpu` do not require quotes and the extra `list` argument can be used with `lsgpu` to remove all objects at once: `rmgpu(list = lsgpu())`.

1.2 Outstanding issues

This release of the R/GPU package is still in beta, therefore there are multiple known issues with the most major ones listed below. The next release(s) of the package will focus on solutions for these known issues. Other issues and bugs will be reported when they are found.

Issues and bugs can be reported on the R/GPU forum or by email, see the forum for more details[1]. Requests and suggestions for new features are very welcome as well as requests for collaboration to further improve the package. Please contact the developers for more details or create a new topic on the forums.

Subsetting of exported data Currently it is not possible to use subsetting on data objects that are exported using the `exporgpu` function, this will result in a warning. Subsetting is only supported on non–exported data objects. Future version of R/GPU will have this feature.

Maximum memory usage When using more GPU memory than what is currently available, R will crash with a CUDA error. The maximum available GPU memory depends on the running window server and active graphical applications. Future versions of R/GPU will include a dynamic check reporting memory usage when exporting data to the GPU memory.

R/GPU precision As can be seen in section 1.6, the choice for single– or double–precision must be made prior to installation of the package. Some of R/GPU's functions perform better in single-precision while other functions have both better performance and higher precision using the double-precision format. For computer systems supporting both precision formats (with NVIDIAs G200 GPU and newer), each R/GPU function will get an extra argument in which the precision can be chosen. This will be available within the next few releases of R/GPU.

Matrices in evalgpu When using matrices with the `evalgpu` function, this matrix will be converted to a vector. The output of `evalgpu` currently is always a vector. This requires the manual conversion from vector to matrix after executing `evalgpu` using R's `matrix` function,

[1] R/GPU forum: `https://gforge.nbic.nl/forum/forum.php?thread_id=188&forum_id=93`

in the following manner: `matrix(evalgpu(x), dim(x))` where `x` is a matrix. The next version of R/GPU is expected to return a proper matrix object when a matrix is used in the expression.

Other issues Currently, not all errors are handled correctly and some may cause the R session to crash. It is therefore advised to always try an R/GPU function with smaller data sets to ensure proper functioning.

1.3 System requirements - Windows

The Windows version of R/GPU requires a CUDA-capable graphics card (see the list of supported graphics cards at the NVIDIA website)[2] and an appropriate graphics driver installed (version for Windows XP >= 162.01, Windows Vista >= 178.08 and for Windows 7 >= 185.85)[3]. It is advised however to always install the most recent version of the NVIDIA graphics card drivers for optimal performance. See section 1.5 for details on obtaining and installing the Windows version of the package. An installation of the CUDA software is not required to use this software.

1.4 System requirements - Linux / Mac OSX

To install this package, the target system must have NVIDIAs CUDA SDK and Toolkit (version >= 2.0) installed as well as a CUDA supported graphics card and appropriate drivers (version >= 169.09). Please refer to the NVIDIA website (see section 1.3) for download links and CUDA installation manuals. To use the double-precision format a GPU with compute capability >= 1.3 is required (*e.g.* G200 series, Quadro FX 5800, Tesla S1070 and C1060).

1.5 Installation Guide - Windows

The Windows binary release can be easily installed from the RGUI through the option "install from local zip-file" or from within an R session using the command `install.packages("rgpu_0.8-1.zip", repos=NULL)`.

Since the Windows release of R/GPU is a binary containing the necessary CUDA DLL files (namely `cudart.DLL` and `cublas.DLL`), please check the R/GPU website regularly for new version whenever a new CUDA version is released. New versions of CUDA often offer higher performance by only loading the newer versions of the DLL files.

[2] Supported GPUs: `http://www.nvidia.com/object/cuda_learn_products.html`
[3] NVIDIA CUDA Download: `http://www.nvidia.com/object/cuda_get.html`

1.6 Installation Guide - Linux / Mac OSX

When installing R/GPU on a Linux or Mac OSX operating system, a few environment variables might need to be set. These variables set the locations of various library and header files required to compile R/GPU. The following three environment variables are required and should be set using the terminal `export` command if the default location is incorrect. The variables are displayed below with the default setting and the command to change them:

- `export CUDA_INC_PATH=/<path_to_cuda_SDK>/common/inc`
  default: `$(HOME)/NVIDIA/CUDA_SDK/common/inc`
- `export CUDA_LIB_PATH=/<path_to_cuda>/lib`
  default: `/usr/local/cuda/lib`
- `export R_INC_PATH=/<path_to_R>/include`
  default: `/usr/local/lib/R/include`

The R/GPU package will be installed with single-precision as default, set the environment variable `RGPU_PRECISION=DOUBLE` prior to installation to use double-precision. As noted in the requirements section, this only works on the most recent GPUs released by NVIDIA.

Download the R/GPU package from our projects website[4] and install the package from within R with the following command:
`install.packages("rgpu_0.8-1.tar.gz", repos=NULL)` or run
`R CMD INSTALL rgpu_0.8-1.tar.gz` from the terminal in the directory containing the downloaded package.

[4] NBIC GeForge R/GPU homepage: `https://gforge.nbic.nl/projects/rgpu/`

Older releases of R/GPU

2 R/GPU version 0.1-2

This is the first public release (alpha) of the R/GPU package.

2.1 Features

Following is a short list describing the features of R/GPU's main functions. See the manual pages for details on how to use these functions (use `?function_name` to see the relevant manual page).

evalgpu This function gives R users the ability to evaluate linear algebra expressions transparently on a CUDA capable Graphics Processing Unit. These expression can comprise of R data objects, unary operators (-), binary operators (+, -, / , *, ^) and parenthesis. The entered – quoted – expression is translated into postfix–notation and transferred to the GPU for evaluation. See the `evalgpu` usage manual in R (with `?evalgpu`) for examples demonstrating the possibilities of this function.

cuda_export The `cuda_export` function can be used to manually export R data objects to the GPU. This can be especially helpful when – some – data is often reused. The exported data objects can only used in an expression for `evalgpu`. Listing the data objects present in the GPU memory is possible with the `cuda_view` function, while removing objects can be done with the `cuda_free` function.

mmult_g Using CUBLAS, this function offers basic matrix multiplication on a GPU (similar to R's %*%command).

2.2 System requirements

This R/GPU release only supports the Linux and Mac OSX operating systems. To use this package, the target system must have NVIDIAs CUDA SDK and Toolkit (version >= 2.0) installed as well as a CUDA supported graphics card and appropriate drivers (version >= 169.09). Please refer to the NVIDIA website (see section 1.3) for download links and CUDA installation manuals.

2.3 Installation guide - Linux / Mac OSX

After successfully installing the CUDA SDK and Toolkit, prior to installing R/GPU, a few environment variables need to be set. These variables set the locations of various library and header files required to compile R/GPU. The following three environment variables should be set using the terminal `export` command as follows:

- `export CUDA_INC_PATH=/<path_to_cuda_SDK>/common/inc`
- `export CUDA_LIB_PATH=/<path_to_cuda>/lib`
- `export R_INC_PATH=/<path_to_R>/include`

Download the R/GPU package from our projects website[5] and install the package from within R with the following command:
`install.packages("rgpu_0.1-2.tar.gz", repos=NULL)` or run
`R CMD INSTALL rgpu_0.1-2.tar.gz` from the terminal in the directory containing the downloaded package.

[5] NBIC GeForge R/GPU homepage: `https://gforge.nbic.nl/projects/rgpu/`