Package 'rgpu'

February 23, 2010

Version 0.8-1

Date 2010-02-12

Title R/GPU: Using the Graphics Processing Unit to speedup bioinformatics analysis with R

Author Marcel Kempenaar <m.kempenaar@med.umcg.nl> dr. Martijn Dijkstra <m.dijkstra@rug.nl>

Maintainer Marcel Kempenaar <m.kempenaar@med.umcg.nl>

Depends R (>= 2.8.0)

OS_type windows

SystemRequirements CUDA-capable GPU, Microsoft Windows XP and newer

Description R/GPU is a user-friendly package that can evaluate any given R expression by making transparent use of an NVIDIA Graphics Processing Unit (GPU) through CUDA. R/GPU can e.g. speedup matrix multiplication with a factor up to 55x and the evaluation of the normal distribution with a factor of up to 9x.

License GPL-3

Website https://gforge.nbic.nl/projects/rgpu/

R topics documented:

`evalgpu` *Evaluates any given R expression by making transparent use of a GPU*

Description

Evaluates linear algebra expressions on the GPU and returns the result. The binary (+, -, *, /, ^) and unary (-) operators are supported on the GPU as well as the functions `exp`, `sin`, `cos`, `log`, `sqrt`, `sum`, `mean`, `%%` (modulo), `%*%` (matrix multiplication) and `%/%` (integer division).

The entered expression may contain constants, vectors and matrices on which the calculation will be done. Objects can be of various lengths and will be reused in the same manner that R does. These objects can be exported to the memory of the GPU prior to evaluation using the `exportgpu` function. This is helpful if data is more often used, reducing overhead caused by memory transfers.

Usage

```
evalgpu(expr, envir = parent.frame(), optimize = TRUE)
```

Arguments

`expr`	R expression to evaluate
`envir`	environment (optional)
`optimize`	logical, enables an optimization step. When set to TRUE `evalgpu` will pre-process the expression identifying constants and will calculate them once on the CPU instead of once for every input element in the expression. This may greatly reduce the number of entities in the expression.

Value

a numerical vector

Warnings

Error handling for data unable to fit into the GPU memory is not yet implemented. Depending on the graphics card and any running graphical applications, the storage capacity available to R/GPU can vary. If the data exceeds the available storage capacity, R will most likely crash.

Currently, objects can also be subsetted but only with data not previously exported.

When using matrices, the resulting vector can be converted into a matrix with the desired dimensions in the following manner: `matrix(evalgpu(x), dim(x))` where `x` is a matrix. Future releases of R/GPU will automatically return the correct output type when using matrices.

See Also

`exportgpu`

Examples

```
## Basic matrix multiplication
A <- matrix(runif(128*256), 128, 256)
B <- matrix(runif(256*64), 256, 64)
evalgpu(A %*% B)

## Calculating the density normal distribution
x = seq(-5, 5, .01); mu = 0; sd = 1
evalgpu(plot( 1 / (sd * sqrt(2 * pi) ) * exp(-( (x - mu)^2) / (2 * sd^2)) ) )

## Correlation
n1 = 1:10; n2 = 1:10 + rnorm(10)
evalgpu(sum( (n1 - mean(n1)) * (n2 - mean(n2)) ) /
sqrt( sum((n1 - mean(n1))^2) * sum((n2 - mean(n2))^2) ))
```

---

exportgpu *Export R object(s) to GPU memory*

Description

Exports numerical R constants, vectors and matrices to GPU memory for use in other R/GPU functions (currently only evalgpu). This function is especially usefull to prevent unnecessary data transfers when algorithms analyze the same data multiple times. The exported data will be copied to device memory where it will reside until freed using the rmgpu function.

Usage

```
exportgpu(...)
```

Arguments

...	objects being exported to the device, comma separated

See Also

rmgpu, lsgpu, getgpudata

Examples

```
## Export R object x
x = runif(1e3)
exportgpu(x)

## Cleanup
rmgpu(list = lsgpu())
```

gemmgpu *Performs matrix multiplication on the GPU*

Description

Performs matrix multiplication on the GPU. Uses a function from the CUBLAS library.

Depending on the used arguments, a full General Matrix Multiply (gemm) operation is possible. The used expression is $C \leftarrow \alpha * op(A) * op(B) + \beta * C$, where $op(X) = X$ or $op(X) = X^T$, scalars α and β, matrices A, B and C.

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.

Usage

```
gemmgpu(A, B, C = "NA",
alpha = 1, beta = 0,
transa = "n", transb = "n")
```

Arguments

`A, B, C`	numerical input matrices (matrix C is optional)
`alpha, beta`	scalars α and β (optional)
`transa, transb`	transpose flag for matrices A and B ('n' or 't') (optional)

Value

`gemmgpu` returns a numerical matrix.

Examples

```
## Generate 2 matrices with random data
A <- matrix(runif(128*256), 128, 256)
B <- matrix(runif(256*64), 256, 64)

## Standard matrix multiplication
C <- gemmgpu(A, B)

## Complete gemm operation with the expression:
## C <- alpha * t(A) %*% t(B) + beta * C
C <- gemmgpu(A, B, C, alpha = 1, beta = 2.5, transa = 't', transb = 't')
```

lsgpu — *Lists all R objects present in the GPU memory*

Description

Lists all R objects present in the GPU memory previously exported using the exportgpu function.

Usage

```
lsgpu()
```

Value

A list containing the names of all objects present in the GPU memory. The output of this function can be used in combination with rmgpu to remove all objects present.

See Also

exportgpu, rmgpu, getgpudata

meangpu — *Calculates the mean using parallel reduction on the GPU*

Description

Generates the mean of input vector x on a CUDA capable device. Uses the reduction example supplied with NVIDIAs CUDA SDK to execute a parallel reduction on the GPU. Supports both vectors and matrices.

Usage

```
meangpu(x)
```

Arguments

x	numeric vector

Value

The sum of the values in "x".

References

M. Harris, *Optimizing parallel reduction in CUDA*, NVidia, Tech. Rep., 2007. http://www.nvidia.com/content/cudazone/cuda_sdk/Data-Parallel_Algorithms.html

See Also

sumgpu

rgpudetails *Shows details on any present CUDA devices*

Description

Shows the available details on the CUDA-capable device(s) present in the system.

Usage

```
rgpudetails()
```

See Also

rgpudevices

rgpudevices *Shows available CUDA devices*

Description

Shows the available CUDA-capable device(s) present in the system with some basic details

Usage

```
rgpudevices(disp)
```

Arguments

disp	default 1 to print device info, 0 to suppress printing (returns number of CUDA devices, used for administrative purposes)

See Also

rgpudetails

rmgpu *Removes exported R objects from GPU memory*

Description

Removes exported R objects using the exportgpu function from GPU memory. Objects can be removed by naming them or using the `list = lsgpu()` functionality.

Usage

```
rmgpu(..., list = character(0))
```

Arguments

...	the objects to be removed, as names (unquoted) or character strings (quoted), comma separated.
list	a list containing names of the objects to be removed.

See Also

exportgpu, lsgpu

Examples

```
## Exporting two R objects to the GPU
x = y = runif(1e3)
exportgpu(x, y)

## Removing previously exported objects
rmgpu(x, y)

## Alternatively, using the lsgpu function to remove all objects
rmgpu(list = lsgpu())
```

---

sumgpu *Summation using parallel reduction on the GPU*

Description

Generates the sum of input vector "x" on a CUDA capable device. Uses the reduction example supplied with NVIDIAs CUDA SDK to execute a parallel reduction on the GPU. Supports both vectors and matrices.

Usage

```
sumgpu(x)
```

Arguments

x	numeric vector

Value

The sum of the values in "x".

References

M. Harris, *Optimizing parallel reduction in CUDA*, NVidia, Tech. Rep., 2007. http://www.nvidia.com/content/cudazone/cuda_sdk/Data-Parallel_Algorithms.html

See Also

meangpu

Index