Package 'clrng'

July 16, 2024

Type Package

Title Parallel Random Number Generation on GPU

Description Builds on 'gpuR' and utilizes the 'clRNG' ('OpenCL') library to provide efficient tools to generate independent random numbers in parallel on a GPU and save the results as 'R' objects, ensuring high-quality random numbers even when 'R' is used interactively or in an ad-hoc manner. Includes Fisher's simulation method adapted from Patefield, William M (1981) <doi:10.2307/2346669> and 'MRG31k3p' Random Number Generator from 'clRNG' library by Advanced Micro Devices, Inc. (2015) <https://github.com/clMathLibraries/clRNG>.

Version 0.0.5

Date 2024-07-11

Maintainer Ruoyong Xu <ruoyong.xu@mail.utoronto.ca>

License GPL-3

Encoding UTF-8

NeedsCompilation yes

Depends R (>= 3.5.0), gpuR

Suggests knitr

LinkingTo RViennaCL, Rcpp, gpuR, RcppEigen

OS_type unix

VignetteBuilder knitr

RoxygenNote 7.3.2

Author Ruoyong Xu [cre, aut, ctb] (<https://orcid.org/0000-0003-3474-8183>), Patrick Brown [aut, ctb] (<https://orcid.org/0000-0003-2541-3744>), Advanced Micro Devices, Inc. [cph] (Copyright holder of clRNG library code)

Repository CRAN

Date/Publication 2024-07-16 09:00:02 UTC

Contents

`createStreamsCpu` *createStreamsCpu*

Description

Create streams stored on the CPU.

Usage

```
createStreamsCpu(n = prod(getOption("clrng.Nglobal")))
```

Arguments

`n`	a integer specifying number of streams to create, default is the number of total work items in use.

Value

an R matrix of streams on CPU.

Examples

```
library(clrng)
if (detectGPUs() >= 1) {
  t(createStreamsCpu(n=5))
  ## GPU streams
  myStreamsGpu = vclMatrix(createStreamsCpu(n=4)) }else {
  message("No GPU context available")
}
```

createStreamsGpu *createStreamsGpu*

Description

Generate streams on a GPU device

Usage

```
createStreamsGpu(n = prod(getOption("clrng.Nglobal")))
```

Arguments

n	a integer specifying number of streams to create, default is the number of total work items in use

Value

a stream object of class 'vclMatrix' on GPU

Examples

```
library(clrng)
if (detectGPUs() >= 1) {
 setBaseCreator(rep(12345,6))
 setContext(grep("gpu", listContexts()$device_type)[1])
 myStreamsGpu = createStreamsGpu(4)
 t(as.matrix(myStreamsGpu))

 myStreamsGpu2 = createStreamsGpu(6)
 t(as.matrix(myStreamsGpu2)) }else {
  message("No GPU context available")
}
```

fisher.sim *fisher.sim*

Description

Performs Monte carlo's simulation for Fisher's exact test on GPU.

Usage

```
fisher.sim(
  x,
  N,
  streams,
  Nglobal = getOption("clrng.Nglobal"),
  type = getOption("clrng.type"),
  returnStatistics = FALSE,
  verbose = FALSE
)
```

Arguments

x	a vclMatrix of integers.
N	an integer specifying number of replicates.
streams	a vclMatrix of streams. Default using streams with package default initial seeds.
Nglobal	a (non-empty) integer vector specifying size of the index space on GPU for use, with default value from global option 'clrng.Nglobal'.
type	a character string specifying "double" or "float" of the returned test statistics, with default value from global option 'clrng.type'.
returnStatistics	a logical value, if TRUE, return test statistics, default is FALSE.
verbose	a logical value, if TRUE, print extra information, default is FALSE.

Value

a 'htest' object of p-value and actual number of replicates and a list of test statistics, streams used, threshold.

Examples

```
library('clrng')
if (detectGPUs() >= 1) {
  setContext(grep("gpu", listContexts()$device_type)[1])

  Job <- matrix(c(1,2,1,0, 3,3,6,1, 10,10,14,9, 6,7,12,11), 4, 4)
  Job <- gpuR::vclMatrix(Job, type="integer")

  getOption('clrng.type')
  options(clrng.Nglobal = c(64,16))
  streams <- createStreamsGpu()
  result <- fisher.sim(Job, 1e5, streams = streams)
  print(result)
  result$streams
  result$threshold
} else {
  message("No GPU context available")
}
```

get_system_info *get_system_info*

Description

This function prints operating system, CPU, GPU, and OpenCL version information.

Usage

```
get_system_info()
```

Value

List of system information.

logfactSum *logfactSum*

Description

Computes the log-factorial of a table on a GPU.

Usage

```
logfactSum(x, Nglobal = getOption("clrng.Nglobal"))
```

Arguments

x	a matrix of integers.
Nglobal	a (non-empty) integer vector specifying size of the index space for use.

Value

sum of log-factorials of elements of the input matrix 'x'.

Examples

```
library(clrng)
if (detectGPUs() >= 1) {
setContext(grep("gpu", listContexts()$device_type)[1])

x <- matrix(c(1:36), 6,6)
logfactSum(x, c(2,2))
# note if matrix is not of integers, a warning will be displayed, eg.
x2 <- matrix(c(1.1,2.1,3.1,4.1,5.1,6.1,7.1,8.1,9.1), 3,3)
is.integer(x2)
```

```
logfactSum(x2, c(16,16))} else {
  message("No GPU context available")
}
```

`month` *Sample dataset: month.RData*

Description

Counts of birth anomalies by birth month and 12 congenital anomalies categories in 2018 within the United States.

Usage

```
data(month)
```

Format

A data frame with 12 rows and 12 columns:

Ane anencephaly

Men meningomyelocele/spina bifida

Cya cyanotic congenital heart disease

Her congenital diaphragmatic hernia

Omp omphalocele

Gas gastrochisis

Lim limb reduction defect

Cle cleft lip with or without cleft palate

Pal cleft palate alone

Dow Down syndrome

Chr suspected chromosomal disorder

Hyp hypospadias.

Source

Centers for Disease Control and Prevention (CDC)

References

https://www.cdc.gov/nchs/data_access/VitalStatsOnline.htm

Examples

```
data(month)
head(month)
```

qqnormGpu *qqnormGpu*

Description

Calculate quantiles on a GPU and produce a Q-Q plot in R.

Usage

```
qqnormGpu(
  y,
  ylim,
  mu = 0,
  sigma = 1,
  lower.tail = 1,
  main = "Normal Q-Q Plot",
  xlab = "Theoretical Quantiles",
  ylab = "Sample Quantiles",
  Nglobal = getOption("clrng.Nglobal"),
  Nlocal = c(2, 2),
  type = getOption("clrng.type"),
  verbose = FALSE,
  ...
)
```

Arguments

y	data sample.
ylim	limits on the plot region.
mu	mean of Normal distribution, default is 0.
sigma	variance of Normal distribution, default is 1.
lower.tail	a logical value specifying whether use lower tail probability, default is TRUE.
main	plot label.
xlab	plot label.
ylab	plot label.
Nglobal	a (non-empty) integer vector specifying size of work items for use, with default value from global option 'clrng.Nglobal'.
Nlocal	a (non-empty) integer vector specifying size of local group size of the index space, default is c(2,2).
type	a character string specifying "double" or "float" of random numbers, with default value from global option 'clrng.type'.
verbose	a logical value, if TRUE, print extra information, default is FALSE.
...	other graphical parameters.

Value

a Normal Q-Q plot.

Examples

```
library(clrng)
if (detectGPUs() >= 1) {
setContext(grep("gpu", listContexts()$device_type)[1])
y <- stats::rt(256, df = 5)  # generates 256 random numbers from a Student's t-distribution
clrng::qqnormGpu(y, ylim=c(-4,5), mu=0, sigma=1, Nglobal=c(16,4))
x<-stats::rnorm(256)
clrng::qqnormGpu(x, Nglobal=c(8,6)) } else {
  message("No GPU context available")
}
```

---

rexpGpu *rexpGpu*

Description

Generate exponential random numbers parallely on a GPU.

Usage

```
rexpGpu(
  n,
  rate = 1,
  streams,
  Nglobal = getOption("clrng.Nglobal"),
  type = getOption("clrng.type"),
  verbose = FALSE
)
```

Arguments

n	a number or a vector specifying the size of the output vector or matrix.
rate	a number specifying the distribution parameter, mean equals to 1/rate.
streams	a vclMatrix of streams.
Nglobal	a (non-empty) integer vector specifying size of work items for use, with default value from global option 'clrng.Nglobal'.
type	a character string specifying "double" or "float" of random numbers, with default value from global option 'clrng.type'.
verbose	a logical value, if TRUE, print extra information, default is FALSE.

Details

type specifies the precision type of random numbers. If GPU supports "double", 'clrng.Nglobal' is "double", otherwise, 'clrng.Nglobal' is "single".

Value

a 'vclVector' or 'vclMatrix' of exponential random numbers.

Examples

```
library('clrng')
if (detectGPUs() >= 1) {
  setContext(grep("gpu", listContexts()$device_type)[1])
  streams <- createStreamsGpu()
  as.vector(rexpGpu(7, streams=streams))

  # produce float precision random numbers
  options(clrng.type='float')
  as.matrix(rexpGpu(c(2,3), rate=0.5, streams))} else {
  message("No GPU context available")
}
```

---

rnormGpu *rnormGpu*

Description

Generate standard Normal random numbers parallely on a GPU.

Usage

```
rnormGpu(
  n,
  streams,
  Nglobal = getOption("clrng.Nglobal"),
  type = getOption("clrng.type"),
  verbose = FALSE
)
```

Arguments

n	a number or a vector specifying the size of output vector or matrix.
streams	a vclMatrix of streams.
Nglobal	a (non-empty) integer vector specifying size of work items for use, with default value from global option 'clrng.Nglobal'.
type	a character string specifying "double" or "float" of random numbers, with default value from global option 'clrng.type'.
verbose	a logical value, if TRUE, print extra information, default is FALSE.

Details

type specifies the precision type of random numbers. If GPU supports "double", 'clrng.Nglobal' is "double", otherwise, 'clrng.Nglobal' is "single".

Value

a 'vclVector' or 'vclMatrix' of standard Normal random numbers.

Examples

```
library(clrng)
if (detectGPUs() >= 1) {
  setContext(grep("gpu", listContexts()$device_type)[1])
  currentPlatform()
  streams <- createStreamsGpu()
  as.vector(rnormGpu(7, streams=streams))

  getOption('clrng.Nglobal')
  # use float precision and global size
  as.matrix(rnormGpu(c(2,3), streams=streams))} else {
  message("No GPU context available")
}
```

---

runifGpu *runifGpu*

Description

Generate uniform random numbers parallely on a GPU.

Usage

```
runifGpu(
  n,
  streams,
  Nglobal = getOption("clrng.Nglobal"),
  type = getOption("clrng.type"),
  verbose = FALSE
)
```

Arguments

n	a number or a numeric vector specifying the size of output vector or matrix.
streams	a vclMatrix of streams.
Nglobal	a (non-empty) integer vector specifying size of work items for use, with default value from global option 'clrng.Nglobal'.

type	a character string specifying "double" or "float" of random numbers, with default value from global option 'clrng.type'.
verbose	a logical value, if TRUE, print extra information, default is FALSE.

Details

type specifies the precision type of random numbers. If GPU supports "double", 'clrng.Nglobal' is "double", otherwise, 'clrng.Nglobal' is "single".

Value

a 'vclVector' or 'vclMatrix' of uniform random numbers.

Examples

```
library('clrng')
if (detectGPUs() >= 1) {
  setContext(grep("gpu", listContexts()$device_type)[1])
  currentDevice()
  getOption('clrng.Nglobal')
  streams <- createStreamsGpu()
  as.vector(runifGpu(5, streams))

  # Change global options
  options(clrng.type="float")
  # produce a matrix of random numbers
  as.matrix(runifGpu(c(2,2), streams))} else {
  message("No GPU context available")
}
```

---

setBaseCreator *setBaseCreator*

Description

Set the initial seed of the first stream.

Usage

```
setBaseCreator(initial = rep(12345, 6))
```

Arguments

initial	a vector of six 31-bit integers specifying the initial state of the first stream. Default is c(12345,12345,12345,12345,12345,12345).

Details

initial is of length 6, recycled if shorter.

Value

a stream object of class 'vclMatrix'.

Examples

```
library('clrng')
setBaseCreator(c(111,222,333,444,555,666))
```

---

week *Sample dataset: week.RData*

Description

Frequencies for birth anomalies by birth day of week and 12 congenital anomalies categories in 2018 within the United States.

Usage

```
data(week)
```

Format

A data frame with 7 rows and 12 columns:

Ane anencephaly

Men meningomyelocele/spina bifida

Cya cyanotic congenital heart disease

Her congenital diaphragmatic hernia

Omp omphalocele

Gas gastrochisis

Lim limb reduction defect

Cle cleft lip with or without cleft palate

Pal cleft palate alone

Dow Down syndrome

Chr suspected chromosomal disorder

Hyp hypospadias.

Source

Centers for Disease Control and Prevention (CDC)

References

https://www.cdc.gov/nchs/data_access/VitalStatsOnline.htm

Index