Package ‘benchmarkme’

November 9, 2019

Type Package
Title Crowd Sourced System Benchmarks
Version 1.0.3
Maintainer Colin Gillespie <csgillespie@gmail.com>
Description Benchmark your CPU and compare against other CPUs.
    Also provides functions for obtaining system specifications, such as
    RAM, CPU type, and R version.
License GPL-2 | GPL-3
URL https://github.com/csgillespie/benchmarkme
BugReports https://github.com/csgillespie/benchmarkme/issues
Depends R (>= 3.5.0)
Imports benchmarkmeData (>= 1.0.3), compiler, doParallel, dplyr,
    foreach, graphics, httr, Matrix, methods, parallel, tibble,
    utils
Suggests covr, DT, ggplot2, knitr, RcppZiggurat, rmarkdown, testthat
VignetteBuilder knitr
Encoding UTF-8
LazyData TRUE
RoxygenNote 6.1.1
NeedsCompilation no
Author Colin Gillespie [aut, cre] (<https://orcid.org/0000-0003-1787-0275>)
Repository CRAN
Date/Publication 2019-11-09 15:10:03 UTC

R topics documented:

benchmarkme-package .................................................. 2
benchmark_io ................................................................ 3
benchmark_std .............................................................. 3
Benchmark your CPU and compare against other CPUs. Also provides functions for obtaining system specifications, such as RAM, CPU type, and R version.

Author(s)

<csgillespie@gmail.com>

See Also

https://github.com/csgillespie/benchmarkme

Examples

```r
## Benchmark your system and compare
## Not run:
res = benchmark_std()
upload_results(res)
plot(res)
```

```r
## End(Not run)
```
benchmark_io

IO benchmarks

Description
Benchmarking reading and writing a csv file (containing random numbers). The tests are essentially `write.csv(x)` and `read.csv(...)` where `x` is a data frame. Of size MB.

Usage
```
benchmark_io(runs = 3, size = c(5, 50), tmpdir = tempdir(),
             verbose = TRUE, cores = 0L)
```
```
bm_read(runs = 3, size = c(5, 50), tmpdir = tempdir(),
         verbose = TRUE)
```
```
bm_write(runs = 3, size = c(5, 50), tmpdir = tempdir(),
          verbose = TRUE)
```

Arguments
- **runs**: Number of times to run the test. Default 3.
- **size**: a number specifying the approximate size of the generated csv. Must be one of 5 or 50
- **tmpdir**: a non-empty character vector giving the directory name. Default `tempdir()`
- **verbose**: Default TRUE.
- **cores**: Default 0 (serial). When cores > 0, the benchmark is run in parallel.

benchmark_std
Run standard benchmarks

Description
This function runs a set of standard benchmarks, which should be suitable for most machines. It runs a collection of matrix benchmark functions

- `benchmark_prog`
- `benchmark_matrix_cal`
- `benchmark_matrix_fun`

To view the list of benchmarks, see `get_available_benchmarks`.

Usage
```
benchmark_std(runs = 3, verbose = TRUE, cores = 0L)
```
Arguments

- **runs** Number of times to run the test. Default 3.
- **verbose** Default TRUE.
- **cores** Default 0 (serial). When cores > 0, the benchmark is run in parallel.

Details

Setting cores equal to 1 is useful for assessing the impact of the parallel computing overhead.

Examples

```r
## Benchmark your system
## Not run:
res = benchmark_std(3)

## Plot results
plot(res)

## End(Not run)
```

Description

A collection of matrix benchmark functions aimed at assessing the calculation speed.

- Creation, transp., deformation of a 2500x2500 matrix.
- 2500x2500 normal distributed random matrix ^1000.
- Sorting of 7,000,000 random values.
- 2500x2500 cross-product matrix (b = a' * a)
- Linear regr. over a 3000x3000 matrix.

These benchmarks have been developed by many authors. See http://r.research.att.com/benchmarks/R-benchmark-25.R for a complete history. The function `benchmark_matrix_cal()` runs the five bm functions.

Usage

```r
bm_matrix_cal_manip(runs = 3, verbose = TRUE)

bm_matrix_cal_power(runs = 3, verbose = TRUE)

bm_matrix_cal_sort(runs = 3, verbose = TRUE)

bm_matrix_cal_cross_product(runs = 3, verbose = TRUE)
```
**bm_matrix_fun_fft**

`bm_matrix_cal_lm(runs = 3, verbose = TRUE)`

`benchmark_matrix_cal(runs = 3, verbose = TRUE, cores = 0L)`

**Arguments**

- **runs**  
  Number of times to run the test. Default 3.
- **verbose**  
  Default TRUE.
- **cores**  
  Default 0 (serial). When cores > 0, the benchmark is run in parallel.

**References**


---

**Description**

A collection of matrix benchmark functions

- FFT over 2,500,000 random values.
- Eigenvalues of a 640x640 random matrix.
- Determinant of a 2500x2500 random matrix.
- Cholesky decomposition of a 3000x3000 matrix.
- Inverse of a 1600x1600 random matrix.

These benchmarks have been developed by many authors. See http://r.research.att.com/benchmarks/R-benchmark-25.R for a complete history. The function `benchmark_matrix_fun()` runs the five `bm` functions.

**Usage**

`bm_matrix_fun_fft(runs = 3, verbose = TRUE)`

`bm_matrix_fun_eigen(runs = 3, verbose = TRUE)`

`bm_matrix_fun_determinant(runs = 3, verbose = TRUE)`

`bm_matrix_fun_cholesky(runs = 3, verbose = TRUE)`

`bm_matrix_fun_inverse(runs = 3, verbose = TRUE)`

`benchmark_matrix_fun(runs = 3, verbose = TRUE, cores = 0L)`
Arguments

- `runs`: Number of times to run the test. Default 3.
- `verbose`: Default TRUE.
- `cores`: Default 0 (serial). When cores > 0, the benchmark is run in parallel.

References


---

**bm_parallel**

Benchmark in parallel

Description

This function runs benchmarks in parallel to test multithreading

Usage

`bm_parallel(bm, runs, verbose, cores, ...)`

Arguments

- `bm`: character name of benchmark function to run from `get_available_benchmarks`
- `runs`: number of runs of benchmark to make
- `verbose`: display messages during benchmarking
- `cores`: number of cores to benchmark. If cores is specified, the benchmark is also run for cores = 1 to allow for normalisation.
- `...`: additional arguments to pass to `bm`

Examples

```r
## Not run:
bm_parallel("bm_matrix_cal_manip", runs = 3, verbose = TRUE, cores = 2)
bm = c("bm_matrix_cal_manip","bm_matrix_cal_power","bm_matrix_cal_sort",
"bm_matrix_cal_cross_product","bm_matrix_cal_lm")
results = lapply(bm, bm_parallel,

## End(Not run)
```
Description

A collection of matrix programming benchmark functions

- 3,500,000 Fibonacci numbers calculation (vector calc).
- Creation of a 3500x3500 Hilbert matrix (matrix calc).
- Grand common divisors of 1,000,000 pairs (recursion).
- Creation of a 1600x1600 Toeplitz matrix (loops).
- Escoufier’s method on a 60x60 matrix (mixed).

These benchmarks have been developed by many authors. See http://r.research.att.com/benchmarks/R-benchmark-25.R for a complete history. The function benchmark_prog() runs the five bm functions.

Usage

bm_prog_fib(runs = 3, verbose = TRUE)
bm_prog_hilbert(runs = 3, verbose = TRUE)
bm_prog_gcd(runs = 3, verbose = TRUE)
bm_prog_toeplitz(runs = 3, verbose = TRUE)
bm_prog_escoufier(runs = 3, verbose = TRUE)
benchmark_prog(runs = 3, verbose = TRUE, cores = 0L)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>runs</td>
<td>Number of times to run the test. Default 3.</td>
</tr>
<tr>
<td>verbose</td>
<td>Default TRUE.</td>
</tr>
<tr>
<td>cores</td>
<td>Default 0 (serial). When cores &gt; 0, the benchmark is run in parallel.</td>
</tr>
</tbody>
</table>
get_available_benchmarks

| create_bundle | Upload benchmark results |

Description

This function uploads the benchmarking results. These results will then be incorporated in future versions of the package.

Usage

create_bundle(results, filename = NULL, args = NULL, id_prefix = "")
upload_results(results, url = "http://www.mas.ncl.ac.uk/~ncsg3/form.php", args = NULL, id_prefix = "")

Arguments

- **results**: Benchmark results. Probably obtained from `benchmark_std()` or `benchmark_io()`.
- **filename**: default NULL. A character vector of where to store the results (in an .rds file). If NULL, results are not saved.
- **args**: Default NULL. A list of arguments to be passed to `get_sys_details()`.
- **id_prefix**: Character string to prefix the benchmark id. Makes it easier to retrieve past results.
- **url**: The location of where to upload the results.

Examples

```r
## Run benchmarks
## Not run:
res = benchmark_std()
upload_results(res)
## End(Not run)
```

get_available_benchmarks

Available benchmarks

Description

The function returns the available benchmarks.

Usage

get_available_benchmarks()
get_byte_compiler

Examples

    get_available_benchmarks()

---

get_byte_compiler  Byte compiler status

Description

Attempts to detect if byte compiling or JIT has been used on the package.

Usage

    get_byte_compiler()

Details

For R 3.5.0 all packages are byte compiled. Before 3.5.0 it was messy. Sometimes the user would
turn it on via JIT, or ByteCompiling the package. On top of that R 3.4.X(?) was byte compiled, but
R 3.4.Y(?) was, not fully optimised!!! What this means is don’t trust historical results!

Value

An integer indicating if byte compiling has been turn on. See ?compiler for details.

Examples

    # Detect if you use byte optimization
    get_byte_compiler()

---

get_cpu  CPU Description

Description

Attempt to extract the CPU model on the current host. This is OS specific:

- Linux: /proc/cpuinfo
- Apple: sysctl -n
- Solaris: Not implemented.
- Windows: wmic cpu

A value of NA is return if it isn’t possible to obtain the CPU.

Usage

    get_cpu()
get_linear_algebra

Get BLAS and LAPACK libraries Extract the blas/lapack from sessionInfo()

Description

Get BLAS and LAPACK libraries Extract the blas/lapack from sessionInfo()

Usage

get_linear_algebra()

get_platform_info

Platform information

Description

This function just returns the output of .Platform

Usage

get_platform_info()

get_ram

Get the amount of RAM

Description

Attempt to extract the amount of RAM on the current machine. This is OS specific:

- Linux: proc/meminfo
- Apple: system_profiler -detailLevel mini
- Windows: memory.size()
- Solaris: prtconf

A value of NA is return if it isn’t possible to determine the amount of RAM.

Usage

get_ram()
References

The print.bytes function was taken from the pryr package.

Examples

```r
#> Return (and pretty print) the amount of RAM
get_ram()
```

<table>
<thead>
<tr>
<th>get_r_version</th>
<th>R version</th>
</tr>
</thead>
</table>

Description

Returns `unclass(R.version)`

Usage

```r
get_r_version()
```

<table>
<thead>
<tr>
<th>get_sys_details</th>
<th>General system information</th>
</tr>
</thead>
</table>

Description

The `get_sys_info` returns general system level information as a list. The function parameters control the information to upload. If a parameter is set to `FALSE`, an `NA` is uploaded instead. Each element of the list contains the output from:

- `Sys.info()`
- `get_platform_info()`
- `get_r_version()`
- `get_ram()`
- `get_cpu()`
- `get_byte_compiler()`
- `get_linear_algebra()`
- `Sys.getlocale()`
- `installed.packages()`
- `.Machine`
- The package version number;
- Unique ID - used to extract results;
- The current date.
Usage

get_sys_details(sys_info = TRUE, platform_info = TRUE,
               r_version = TRUE, ram = TRUE, cpu = TRUE, byte_compiler = TRUE,
               linear_algebra = TRUE, locale = TRUE, installed_packages = TRUE,
               machine = TRUE)

Arguments

sys_info          Default TRUE.
platform_info     Default TRUE.
r_version         Default TRUE.
ram               Default TRUE.
cpu               Default TRUE.
byte_compiler     Default TRUE.
linear_algebra    Default TRUE.
locale            Default TRUE.
installed_packages Default TRUE.
machine           Default TRUE.

Value

A list

Examples

## Returns all details about your machine
get_sys_details()

plot.ben_results  Compare results to past tests

Description

Plotting

Usage

## S3 method for class 'ben_results'
plot(x, test_group = unique(x$test_group),
      blas_optimize = is_blas_optimize(x), log = "y", ...)

 getDataFromBenResults

## S3 method for class 'getDataFromBenResults'
plot(x, test_group = unique(x$test_group),
      blas_optimize = is_blas_optimize(x), log = "y", ...)

getDataFromBenResults
**Arguments**

- **x**: The output from a benchmark_* call.
- **test_group**: Default `unique(x$test_group)`. The default behaviour is select the groups from your benchmark results.
- **blas_optimize**: Logical. Default The default behaviour is to compare your results with results that use the same blas_optimize setting. To use all results, set to `NULL`.
- **log**: By default the y axis is plotted on the log scale. To change, set the the argument equal to the empty parameter string, "".
- **...**: Arguments to be passed to other downstream methods.

**Examples**

```r
data(sample_results)
plot(sample_results, blas_optimize = NULL)
```

**Description**

Comparison with past results.

**Usage**

```r
rank_results(results, blas_optimize = is_blas_optimize(results),
             verbose = TRUE)
```

**Arguments**

- **results**: Benchmark results. Probably obtained from benchmark_std() or benchmark_io().
- **blas_optimize**: Logical. Default The default behaviour is to compare your results with results that use the same blas_optimize setting. To use all results, set to `NULL`.
- **verbose**: Default TRUE.

**Sample results**

Sample benchmarking results

**Format**

A data frame
Index

*Topic** package
  benchmarkme-package, 2

benchmark_io, 3
benchmark_matrix_cal
  (bm_matrix_cal_manip), 4
benchmark_matrix_fun
  (bm_matrix_fun_fft), 5
benchmark_prog (bm_prog_fib), 7
benchmark_std, 3
benchmarkme (benchmarkme-package), 2
benchmarkme-package, 2
bm_matrix_cal_cross_product
  (bm_matrix_cal_manip), 4
bm_matrix_cal_lm (bm_matrix_cal_manip), 4
bm_matrix_cal_manip, 4
bm_matrix_cal_power
  (bm_matrix_cal_manip), 4
bm_matrix_cal_sort
  (bm_matrix_cal_manip), 4
bm_matrix_fun_cholesky
  (bm_matrix_fun_fft), 5
bm_matrix_fun_determinant
  (bm_matrix_fun_fft), 5
bm_matrix_fun_eigen
  (bm_matrix_fun_fft), 5
bm_matrix_fun_fft, 5
bm_matrix_fun_inverse
  (bm_matrix_fun_fft), 5
bm_parallel, 6
bm_prog_escoufier (bm_prog_fib), 7
bm_prog_fib, 7
bm_prog_gcd (bm_prog_fib), 7
bm_prog_hilbert (bm_prog_fib), 7
bm_prog_toeplitz (bm_prog_fib), 7
bm_read (benchmark_io), 3
bm_write (benchmark_io), 3
create_bundle, 8

get_available_benchmarks, 6, 8
get_byte_compiler, 9
get_cpu, 9
get_linear_algebra, 10
get_platform_info, 10
get_r_version, 11
get_ram, 10
get_sys_details, 11
plot.ben_results, 12
rank_results, 13
sample_results, 13
upload_results (create_bundle), 8