A Machine Profile Summary

John C. Nash, University of Ottawa, Canada

08/06/2021

Providing a characterization of a particular computing environment for R calculations

I have looked at this.

If we want to compare results computed on different machines, we need a way to provide measures and identifiers of the particular computing environment at hand. This document attempts to summarize some possibilities available in the R statistical software and language.

Some possible information desired

There are multiple ways to get information of the type useful for characterizing the computing infrastructure used for a particular test or timing. We will need to ensure that identifiers of particular pieces of information are unique, and that common identifiers really do specify identical information. Otherwise, we need to develop new names for our set of information.

Temporal information

Most computatins should have a time/date stamp that is unlikely to be confused with any other, at least in combination with a machine name and/or other tags.

```
#get a timestamp
tsstr<- format(Sys.time(), "%Y%m%d%H%M") # tsstr == time stamp string in form YYYYmmddHHMM
cat("Date and Time stamp:",tsstr,"\n")
```

```
## Date and Time stamp: 202106151258
```

Identifier for the computing environment

Some of the important information elements concerning the computing environment that we need for reporting tests are

- machine name. While most systems (and Linux in particular) offer to let the user provide a machine name, there are generally defaults that many people accept, and these are often uninformative and may be non-unique. We note that VirtualBox with a Windows 10 guest machine gave the name DESKTOP-HF4CKVA. Where this name was generated we are not sure. Settings / System allows "rename this PC".
- operating system and version
- compiler or interpreter version. For our needs, it is obvious that the R version will be important. However, if any code is compiled or linked to libraries, we would like to know that. In particular, versions of compilers (e.g., gfortran, gcc) or BLAS or LAPACK libraries used will affect performance. Linux generally displays the BLAS and LAPACK **filenames** in `sessionInfo()`, but to get the version

information, one needs to dig deeper, for example, using operating system commands. For the present, we will omit such extra information unless we can extract it easily within R.

- specific variations, if any, that should be noted. Here we may want to note if a machine is running background tasks, or if some special steps have been taken to speed up or slow down the operation e.g., overclocking.

Hardware information

Some of the factors relating to hardware that could be important are:

- cpu
- cores
- operating cycle speed (While the cpu is generally specified)
- RAM total
- RAM available, possibly indicating swap space and usage
- RAM speed (could be variable)

Software information

?? what to put in.

R tools for machine information

sessionInfo()

```
si <- sessionInfo()
si <- as.vector(si)
si
```

```
## R version 4.1.0 (2021-05-18)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Linux Mint 20.1
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
## LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/liblapack.so.3
##
## locale:
##  [1] LC_CTYPE=en_CA.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_CA.UTF-8        LC_COLLATE=en_CA.UTF-8
##  [5] LC_MONETARY=en_CA.UTF-8    LC_MESSAGES=en_CA.UTF-8
##  [7] LC_PAPER=en_CA.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_CA.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.1.0    magrittr_2.0.1    tools_4.1.0       htmltools_0.5.1.1
##  [5] yaml_2.2.1        stringi_1.6.2     rmarkdown_2.8     knitr_1.33
##  [9] stringr_1.4.0     xfun_0.23         digest_0.6.27     rlang_0.4.11
## [13] evaluate_0.14
```

benchmarkme

- uses `proc.time()` for timings
- has single core and parallel timings
- ?? does it do GPUs
- provides far too much for most needs i.e., a succinct summary of how capable a computing system should be

```
library(benchmarkmeData)
library(benchmarkme)
ls(package:benchmarkmeData)
```

```
## Warning in ls(package:benchmarkmeData): 'package:benchmarkmeData' converted to
## character string
```

```
## [1] "get_datatable_past" "is_blas_optimize"   "make_data_set"
## [4] "move_files"         "past_results"       "past_results_v2"
## [7] "plot_past"          "select_results"     "summarise_results"
```

```
ls(package:benchmarkme)
```

```
## Warning in ls(package:benchmarkme): 'package:benchmarkme' converted to character
## string
```

```
##  [1] "benchmark_io"                "benchmark_matrix_cal"
##  [3] "benchmark_matrix_fun"        "benchmark_prog"
##  [5] "benchmark_std"               "bm_matrix_cal_cross_product"
##  [7] "bm_matrix_cal_lm"            "bm_matrix_cal_manip"
##  [9] "bm_matrix_cal_power"         "bm_matrix_cal_sort"
## [11] "bm_matrix_fun_cholesky"      "bm_matrix_fun_determinant"
## [13] "bm_matrix_fun_eigen"         "bm_matrix_fun_fft"
## [15] "bm_matrix_fun_inverse"       "bm_parallel"
## [17] "bm_prog_escoufier"           "bm_prog_fib"
## [19] "bm_prog_gcd"                 "bm_prog_hilbert"
## [21] "bm_prog_toeplitz"            "bm_read"
## [23] "bm_write"                    "create_bundle"
## [25] "get_available_benchmarks"    "get_byte_compiler"
## [27] "get_cpu"                     "get_linear_algebra"
## [29] "get_platform_info"           "get_r_version"
## [31] "get_ram"                     "get_sys_details"
## [33] "is_blas_optimize"            "plot_past"
## [35] "rank_results"                "sample_results"
## [37] "upload_results"
```

```
lsf.str("package:benchmarkme")
```

```
## benchmark_io : function (runs = 3, size = c(5, 50), tmpdir = tempdir(), verbose = TRUE,
##     cores = 0L)
## benchmark_matrix_cal : function (runs = 3, verbose = TRUE, cores = 0L)
## benchmark_matrix_fun : function (runs = 3, verbose = TRUE, cores = 0L)
## benchmark_prog : function (runs = 3, verbose = TRUE, cores = 0L)
## benchmark_std : function (runs = 3, verbose = TRUE, cores = 0L)
## bm_matrix_cal_cross_product : function (runs = 3, verbose = TRUE)
## bm_matrix_cal_lm : function (runs = 3, verbose = TRUE)
## bm_matrix_cal_manip : function (runs = 3, verbose = TRUE)
## bm_matrix_cal_power : function (runs = 3, verbose = TRUE)
## bm_matrix_cal_sort : function (runs = 3, verbose = TRUE)
## bm_matrix_fun_cholesky : function (runs = 3, verbose = TRUE)
```

```
## bm_matrix_fun_determinant : function (runs = 3, verbose = TRUE)
## bm_matrix_fun_eigen : function (runs = 3, verbose = TRUE)
## bm_matrix_fun_fft : function (runs = 3, verbose = TRUE)
## bm_matrix_fun_inverse : function (runs = 3, verbose = TRUE)
## bm_parallel : function (bm, runs, verbose, cores, ...)
## bm_prog_escoufier : function (runs = 3, verbose = TRUE)
## bm_prog_fib : function (runs = 3, verbose = TRUE)
## bm_prog_gcd : function (runs = 3, verbose = TRUE)
## bm_prog_hilbert : function (runs = 3, verbose = TRUE)
## bm_prog_toeplitz : function (runs = 3, verbose = TRUE)
## bm_read : function (runs = 3, size = c(5, 50), tmpdir = tempdir(), verbose = TRUE)
## bm_write : function (runs = 3, size = c(5, 50), tmpdir = tempdir(), verbose = TRUE)
## create_bundle : function (results, filename = NULL, args = NULL, id_prefix = "")
## get_available_benchmarks : function ()
## get_byte_compiler : function ()
## get_cpu : function ()
## get_linear_algebra : function ()
## get_platform_info : function ()
## get_r_version : function ()
## get_ram : function ()
## get_sys_details : function (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)
## is_blas_optimize : function (results)
## plot_past : function (test_group, blas_optimize = NULL, cores = 0, log = "y")
## rank_results : function (results, blas_optimize = is_blas_optimize(results), verbose = TRUE)
## upload_results : function (results, url = "http://www.mas.ncl.ac.uk/~ncsg3/form.php", args = NULL,
##     id_prefix = "")
```

```
lsf.str("package:benchmarkmeData")
```

```
## get_datatable_past : function (test_group, blas_optimize = NULL, cores = 0)
## is_blas_optimize : function (results)
## make_data_set : function (from)
## move_files : function (from, to)
## plot_past : function (test_group, blas_optimize = NULL, cores = 0, log = "y")
## select_results : function (test_group, results = NULL, blas_optimize = NULL, cores = 0)
## summarise_results : function (res)
```

```
## This next line takes a lot of time to run, so is commented out here
# benchmark_std()
get_byte_compiler()
```

```
## byte_optimize
##             2
```

```
gla<-get_linear_algebra()
gsd<-get_sys_details()
```

calceps.R

This is a port of Mike Malcolm's ENVRON to R. ??ref. It computes the machine precision, radix and number of radix digits.

?? get file into chunk

`calceps.R`

- calceps.R example

Sys.info()

```r
Sys.info()
```

```
##                                       sysname
##                                       "Linux"
##                                       release
##                               "5.4.0-74-generic"
##                                       version
## "#83-Ubuntu SMP Sat May 8 02:35:39 UTC 2021"
##                                      nodename
##                                          "M21"
##                                       machine
##                                      "x86_64"
##                                         login
##                                     "unknown"
##                                          user
##                                       "john"
##                                 effective_user
##                                       "john"
```

Issues relateing to RNG

Issues relating to compilation of R

- LAPACK
- BLAS
- others?

Tools for accessing and clearing environments and dataframes

?? should this be here – probably to list things in the workspace

```
sys.frame()
sys.frames()
sys.status()
sys.on.exit()
sys.parents()
sys.calls()
```

??JN exp(sin(cos())) benchmark

Choices

For use in recording tests of R functions and packages for optimization and nonlinear least squares, it seems that the `benchmarkme` function `get_sys_details()` provides more than sufficient information for our needs.

From the above discussion, the following offers a possible compact solution.

```
sy<-Sys.info()
tsstr<- format(Sys.time(), "%Y%m%d%H%M")
cpu<-benchmarkme::get_cpu()
ram<-benchmarkme::get_ram()
machid<-paste(sy["nodename"],":",sy["user"],"-",sy["sysname"],"-",sy["release"],
              "|",cpu$model_name,"|",ram,"bytesRAM", sep='')
cat(machid,"\n")
```

```
## M21:john-Linux-5.4.0-74-generic|Intel(R) Core(TM) i5-10400 CPU @ 2.90GHz|33484775424bytesRAM
```