Package: syrup (via r-universe)

October 16, 2024

Title Measure Memory and CPU Usage for Parallel R Code

Version 0.1.1.9000

Description Measures memory and CPU usage of R code by regularly taking snapshots of calls to the system command 'ps'. The package provides an entry point (albeit coarse) to profile usage of system resources by R code run in parallel.

License MIT + file LICENSE

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Depends bench

Imports callr, dplyr, ps, purrr, rlang, tibble, vctrs, withr

Config/Needs/website rmarkdown

URL https://github.com/simonpcouch/syrup, https://simonpcouch.github.io/syrup/

BugReports https://github.com/simonpcouch/syrup/issues

Repository https://simonpcouch.r-universe.dev

RemoteUrl https://github.com/simonpcouch/syrup

RemoteRef HEAD

RemoteSha 38223f782498e4c756fb438a4fa9f106a0ff11be

Contents

syrup *Memory and CPU Usage Information for Parallel R Code*

Description

This function is a wrapper around the system command `ps` that can be used to benchmark (peak) memory and CPU usage of parallel R code. By taking snapshots the memory usage of R processes at a regular `interval`, the function dynamically builds up a profile of their usage of system resources.

Usage

```
syrup(expr, interval = 0.5, peak = FALSE, env = caller_env())
```

Arguments

`expr`	An expression.
`interval`	The interval at which to take snapshots of respirce usage. In practice, there's an overhead on top of each of these intervals.
`peak`	Whether to return rows for only the "peak" memory usage. Interpreted as the `id` with the maximum `rss` sum. Defaults to `FALSE`, but may be helpful to set `peak = TRUE` for potentially very long-running processes so that the tibble doesn't grow too large.
`env`	The environment to evaluate `expr` in.

Details

While much of the verbiage in the package assumes that the supplied expression will be distributed across CPU cores, there's nothing specific about this package that necessitates the expression provided to `syrup()` is run in parallel. Said another way, `syrup()` will work just fine with "normal," sequentially-run R code (as in the examples). That said, there are many better, more fine-grained tools for the job in the case of sequential R code, such as `Rprofmem()`, the profmem package, the bench package, and packages in the R-prof GitHub organization.

Loosely, the function works by:

- Setting up another R process (call it `sesh`) that queries system information using `ps::ps()` at a regular interval,
- Evaluating the supplied expression,
- Reading the queried system information back into the main process from `sesh`,
- Closing `sesh`, and then
- Returning the queried system information.

Note that information on the R process `sesh` is filtered out from the results automatically.

Value

A tibble with columns `id` and `time` and a number of columns from `ps::ps()` output describing memory and CPU usage. Notably, the process ID `pid`, parent process ID `ppid`, percent CPU usage, and resident set size `rss` (a measure of memory usage).

Examples

```
# pass any expression to syrup. first, sequentially:
res_syrup <- syrup({res_output <- Sys.sleep(1)})

res_syrup

# to snapshot memory and CPU information more (or less) often, set `interval`
syrup(Sys.sleep(1), interval = .01)

# use `peak = TRUE` to return only the snapshot with
# the highest memory usage (as `sum(rss)`)
syrup(Sys.sleep(1), interval = .01, peak = TRUE)

# results from syrup are more---or maybe only---useful when
# computations are evaluated in parallel. see package README
# for an example.
```

Index