Package ‘data.table.threads’

May 8, 2026

Title Analyze Multi-Threading Performance for 'data.table' Functions

Version 1.0.1

Description Assists in finding the most suitable thread count for the various 'data.table' routines that support parallel processing.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.1

URL https://github.com/Anirban166/data.table.threads

Imports ggplot2, data.table, microbenchmark

NeedsCompilation no

Author Anirban Chetia [aut, cre]

Maintainer Anirban Chetia <ac4743@nau.edu>

Repository CRAN

Date/Publication 2024-11-10 16:50:02 UTC

Contents

addRecommendedEfficiency 2
findOptimalThreadCount 3
plot.data_table_threads_benchmark 4
print.data_table_threads_benchmark 5
runBenchmarks 6
setThreadCount 6

Index **8**

`addRecommendedEfficiency`

Function that adds recommended efficiency speedup lines and points to benchmarks

Description

This function adds to the timing results (or the benchmarked data). It computes the recommended efficiency speedup line and the point which denotes the recommended thread count, both being based on the specified efficiency value.

Usage

```
addRecommendedEfficiency(benchmarkData, recommendedEfficiency = 0.5)
```

Arguments

`benchmarkData` A `data.table` of class `data_table_threads_benchmark` containing benchmarked results, which includes timings and speedup plot data (ideal and measured types) for each function.

`recommendedEfficiency` A numeric value between 0 and 1 that defines the slope for the "Recommended" efficiency speedup line. (Default is 0.5)

Details

This function allows users to add a "Recommended" efficiency line to previously computed benchmark data (without needing to recompute the timings). The recommended speedup is based on the provided efficiency value, which adjusts the slope of the speedup curve and correspondingly helps in the computation of the closest point of measured speedup to the "Recommended" speedup curve.

Value

The input `data.table` with the recommended efficiency added to the plot data (attributes).

See Also

`findOptimalThreadCount` for computing the benchmark data with measured and ideal speedup data.

Examples

```
# Finding the best performing thread count for each benchmarked data.table function
# with a data size of 1000 rows and 10 columns:
benchmarks <- data.table.threads::findOptimalThreadCount(1e3, 10)
# Adding recommended efficiency to the plot data:
addRecommendedEfficiency(benchmarks, recommendedEfficiency = 0.6)
```

`findOptimalThreadCount` *Function that finds the optimal (fastest) thread count for different* `data.table` *functions*

Description

This function finds the optimal thread count for running `data.table` functions with maximum efficiency.

Usage

```
findOptimalThreadCount(rowCount, colCount, times = 10, verbose = FALSE)
```

Arguments

`rowCount`	The number of rows in the `data.table`.
`colCount`	The number of columns in the `data.table`.
`times`	The number of times the benchmarks are to be run.
`verbose`	Option (logical) to enable or disable detailed message printing.

Details

Iteratively runs benchmarks with increasing thread counts and determines the optimal number of threads for each `data.table` function.

Value

A `data.table` of class `data_table_threads_benchmark` containing the optimal thread count for each `data.table` function.

Examples

```
# Finding the best performing thread count for each benchmarked data.table function
# with a data size of 1000 rows and 10 columns:
(optimalThreads <- data.table.threads::findOptimalThreadCount(1e3, 10))
```

`plot.data_table_threads_benchmark` *Function to make speedup plots for the benchmarked* `data.table` *functions*

Description

Function to make speedup plots for the benchmarked `data.table` functions

Usage

```
## S3 method for class 'data_table_threads_benchmark'
plot(x, ...)
```

Arguments

`x`	A `data.table` of class `data_table_threads_benchmark` containing benchmarked timings with corresponding thread counts.
`...`	Additional arguments (not used in this function but included for consistency with the S3 generic `plot` function).

Details

Creates a comprehensive `ggplot` showing the ideal, sub-optimal, and measured speedup trends for the `data.table` functions benchmarked with varying thread counts.

Value

A `ggplot` object containing a speedup plot for each benchmarked `data.table` function.

Examples

```
# Finding the best performing thread count for each benchmarked data.table function
# with a data size of 1000 rows and 10 columns:
benchmarkData <- data.table.threads::findOptimalThreadCount(1e3, 10)
# Generating speedup plots based on the data collected above:
plot(benchmarkData)
```

`print.data_table_threads_benchmark` *Function to concisely display the results returned by* `findOptimalThreadCount()` *in an organized table*

Description

Function to concisely display the results returned by `findOptimalThreadCount()` in an organized table

Usage

```
## S3 method for class 'data_table_threads_benchmark'
print(x, ...)
```

Arguments

`x`	A `data.table` of class `data_table_threads_benchmark` containing benchmarked timings with corresponding thread counts.
`...`	Additional arguments (not used in this function but included for consistency with the S3 generic `print` function).

Details

Prints a table enlisting the best performing thread count along with the runtime (median value) for each benchmarked `data.table` function.

Value

NULL.

Examples

```
# Finding the best performing thread count for each benchmarked data.table function
# with a data size of 1000 rows and 10 columns:
(benchmarkData <- data.table.threads::findOptimalThreadCount(1e3, 10))
```

runBenchmarks — *Function to run a set of predefined benchmarks for different* `data.table` *functions with varying thread counts*

Description

Function to run a set of predefined benchmarks for different `data.table` functions with varying thread counts

Usage

```
runBenchmarks(rowCount, colCount, threadCount, times = 10, verbose = TRUE)
```

Arguments

`rowCount`	The number of rows in the `data.table`.
`colCount`	The number of columns in the `data.table`.
`threadCount`	The total number of threads to use.
`times`	The number of times the benchmarks are to be run.
`verbose`	Option (logical) to enable or disable detailed message printing.

Details

Benchmarks various `data.table` functions that are parallelizable (`setorder`, `GForce_sum`, `subsetting`, `frollmean`, `fcoalesce`, `between`, `fifelse`, `nafill`, and `CJ`) with varying thread counts.

Value

A `data.table` containing benchmarked timings for each `data.table` function with different thread counts.

setThreadCount — *Function to set the thread count for a specific* `data.table` *function*

Description

Function to set the thread count for a specific `data.table` function

Usage

```
setThreadCount(
  benchmarkData,
  functionName,
  efficiencyFactor = 0.5,
  verbose = FALSE
)
```

Arguments

`benchmarkData` A `data.table` of class `data_table_threads_benchmark` containing benchmarked timings with corresponding thread counts.

`functionName` The name of the `data.table` function for which to set the thread count.

`efficiencyFactor` A numeric value between 0 and 1 indicating the desired efficiency level for thread count selection. 0 represents use of the optimal thread count (lowest median runtime) and 0.5 represents the recommended thread count.

`verbose` Option (logical) to enable or disable detailed message printing.

Details

Sets the thread count to either the optimal (fastest median runtime) or recommended value (default) based on the chosen type argument for the specified `data.table` function based on the results obtained from `findOptimalThreadCount()`.

Value

NULL.

Examples

```
# Finding the best performing thread count for each benchmarked data.table function
# with a data size of 1000 rows and 10 columns:
benchmarkData <- data.table.threads::findOptimalThreadCount(1e3, 10)
# Setting the optimal thread count for the 'forder' function:
setThreadCount(benchmarkData, "forder", efficiencyFactor = 1)
# Can verify by checking benchmarkData and getDTthreads():
data.table::getDTthreads()
```

Index

addRecommendedEfficiency, 2

findOptimalThreadCount, 2, 3

plot.data_table_threads_benchmark, 4
print.data_table_threads_benchmark, 5

runBenchmarks, 6

setThreadCount, 6