

# Package ‘data.table.threads’

October 11, 2024

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

**Version** 1.0.0

**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** 2024-10-11 08:40:02 UTC

## Contents

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

<b>Index</b>	<b>7</b>
--------------	----------

---

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

<code>x</code>	A <code>data.table</code> of class <code>data_table_threads_benchmark</code> containing benchmarked timings with corresponding thread counts.
<code>...</code>	Additional arguments (not used in this function but included for consistency with the S3 generic <code>print</code> 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	<i>Function to run a set of predefined benchmarks for different data.table functions with varying thread counts</i>
---------------	---

---

**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	<i>Function to set the thread count for a specific data.table function</i>
----------------	--

---

**Description**

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

**Usage**

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

**Arguments**

<code>benchmarkData</code>	A <code>data.table</code> of class <code>data_table_threads_benchmark</code> containing benchmarked timings with corresponding thread counts.
<code>functionName</code>	The name of the <code>data.table</code> function for which to set the thread count.
<code>efficiencyFactor</code>	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.
<code>verbose</code>	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

`findOptimalThreadCount`, [2](#)

`plot.data_table_threads_benchmark`, [3](#)

`print.data_table_threads_benchmark`, [4](#)

`runBenchmarks`, [5](#)

`setThreadCount`, [5](#)