Package 'clustermq'

September 29, 2018

Title Evaluate Function Calls on HPC Schedulers (LSF, SGE, SLURM, PBS/Torque)

Version 0.8.5

Author Michael Schubert <mschu.dev@gmail.com>

Maintainer Michael Schubert <mschu.dev@gmail.com>

Description Evaluate arbitrary function calls using workers on HPC schedulers in single line of code. All processing is done on the network without accessing the file system. Remote schedulers are supported via SSH.

URL https://github.com/mschubert/clustermq

BugReports https://github.com/mschubert/clustermq/issues

Depends R (>= 3.0.2)

Imports infuser (>= 0.2.8), narray, progress, purrr, R6, rzmq (>= 0.9.4), utils

License Apache License (== 2.0) | file LICENSE

LazyData true

Encoding UTF-8

Suggests dplyr, foreach, iterators, knitr, parallel, roxygen2 (>= 5.0.0), testthat, tools

VignetteBuilder knitr

RoxygenNote 6.1.0.9000

NeedsCompilation no

Repository CRAN

Date/Publication 2018-09-29 13:40:03 UTC

R topics documented:

`.onAttach` *Report queueing system on package attach if not set*

Description

Report queueing system on package attach if not set

Usage

```
.onAttach(libname, pkgname)
```

Arguments

libname	default arg for compatibility
pkgname	default arg for compatibility

.onLoad *Select the queueing system on package loading*

Description

This is done by setting the variable 'qsys' in the package environment to the object that contains the desired queueing system. We further call its setup() function if it exists, and set the variable 'qsys_id' to the scheduler we use

Usage

```
.onLoad(libname, pkgname)
```

Arguments

libname	default arg for compatibility
pkgname	default arg for compatibility

---

bind_avail *Binds an rzmq to an available port in given range*

Description

Binds an rzmq to an available port in given range

Usage

```
bind_avail(socket, range, iface = "tcp://*", n_tries = 100)
```

Arguments

socket	An rzmq socket object
range	Numbers to consider (e.g. 6000:8000)
iface	Interface to listen on
n_tries	Number of ports to try in range

Value

The port the socket is bound to

check_args *Function to check arguments with which Q() is called*

Description

Function to check arguments with which Q() is called

Usage

```
check_args(fun, iter, const = list())
```

Arguments

fun	A function to call
iter	Objects to be iterated in each function call
const	A list of constant arguments passed to each function call

Value

Processed iterated argument list if 'iter' is a list

chunk *Subset index chunk for processing*

Description

'attr' in '[.data.frame' takes too much CPU time

Usage

```
chunk(x, i)
```

Arguments

x	Index data.frame
i	Rows to subset

Value

x[i,]

clustermq *Evaluate Function Calls on HPC Schedulers (LSF, SGE, SLURM)*

Description

Provides the `Q` function to send arbitrary function calls to workers on HPC schedulers without relying on network-mounted storage. Allows using remote schedulers via SSH.

Details

Under the hood, this will submit a cluster job that connects to the master via TCP the master will then send the function and argument chunks to the worker and the worker will return the results to the master until everything is done and you get back your result

Computations are done entirely on the network and without any temporary files on network-mounted storage, so there is no strain on the file system apart from starting up R once per job. This removes the biggest bottleneck in distributed computing.

Using this approach, we can easily do load-balancing, i.e. workers that get their jobs done faster will also receive more function calls to work on. This is especially useful if not all calls return after the same time, or one worker has a high load.

For more detailed usage instructions, see the documentation of the `Q` function.

cmq_foreach *clustermq foreach handler*

Description

clustermq foreach handler

Usage

```
cmq_foreach(obj, expr, envir, data)
```

Arguments

`obj`	Returned from foreach::foreach, containing the following variables: args : Arguments passed, each as a call argnames: character vector of arguments passed evalenv : Environment where to evaluate the arguments export : character vector of variable names to export to nodes packages: character vector of required packages verbose : whether to print status messages [logical] errorHandling: string of function name to call error with, e.g. "stop"
`expr`	An R expression in curly braces
`envir`	Environment where to evaluate the arguments
`data`	Common arguments passed by register_dopcar_cmq(), e.g. n_jobs

DO_CHUNK *Chunk of iterated arguments for the worker*

Description

Field has to be ‘chunk‘

Usage

```
DO_CHUNK
```

Format

An object of class `integer` of length 1.

DO_SETUP *Message contains common data*

Description

This includes the function definition, common data, export

Usage

```
DO_SETUP
```

Format

An object of class `integer` of length 1.

host *Construct the ZeroMQ host*

Description

Construct the ZeroMQ host

Usage

```
host(short = getOption("clustermq.short.host", TRUE))
```

Arguments

`short`	whether to use unqualified host name (before first dot)

Value

the host name as character string

`LOCAL` *Placeholder for local processing*

Description

Mainly so tests pass without setting up a scheduler

Usage

```
LOCAL
```

Format

An object of class `R6ClassGenerator` of length 24.

`LSF` *LSF scheduler functions*

Description

Derives from QSys to provide LSF-specific functions

Usage

```
LSF
```

Format

An object of class `R6ClassGenerator` of length 24.

master *Master controlling the workers*

Description

exchanging messages between the master and workers works the following way: * we have submitted a job where we don't know when it will start up * it starts, sends is a message list(id=0) indicating it is ready * we send it the function definition and common data * we also send it the first data set to work on * when we get any id > 0, it is a result that we store * and send the next data set/index to work on * when computatons are complete, we send id=0 to the worker * it responds with id=-1 (and usage stats) and shuts down

Usage

```
master(qsys, iter, rettype = "list", fail_on_error = TRUE,
  chunk_size = NA, timeout = Inf)
```

Arguments

qsys	Instance of QSys object
iter	Objects to be iterated in each function call
rettype	Return type of function
fail_on_error	If an error occurs on the workers, continue or fail?
chunk_size	Number of function calls to chunk together defaults to 100 chunks per worker or max. 500 kb per chunk
timeout	Maximum time in seconds to wait for worker (default: Inf)

Value

A list of whatever 'fun' returned

MULTICORE *Process on multiple cores on one machine*

Description

This makes use of rzmq messaging and sends requests via TCP/IP

Usage

```
MULTICORE
```

Format

An object of class `R6ClassGenerator` of length 24.

PROXY_CMD *Message is an SSH command*

Description

Field is either ‘exec‘ (command to run) or ‘reply‘ (how it went)

Usage

```
PROXY_CMD
```

Format

An object of class `integer` of length 1.

PROXY_READY *Message ID indicating SSH proxy is ready to distribute data*

Description

Field has to be ‘proxy‘

Usage

```
PROXY_READY
```

Format

An object of class `integer` of length 1.

PROXY_STOP *Message telling the SSH proxy to clean up*

Description

No fields. Signals the worker to break its main loop.

Usage

```
PROXY_STOP
```

Format

An object of class `integer` of length 1.

PROXY_UP *Message ID indicating SSH proxy is up*

Description

Message ID indicating SSH proxy is up

Usage

```
PROXY_UP
```

Format

An object of class `integer` of length 1.

purrr_lookup *Lookup table for return types to purrr functions*

Description

Lookup table for return types to purrr functions

Usage

```
purrr_lookup
```

Format

An object of class `list` of length 9.

Q *Queue function calls on the cluster*

Description

Queue function calls on the cluster

Usage

```
Q(fun, ..., const = list(), export = list(), seed = 128965,
  memory = NULL, template = list(), n_jobs = NULL, job_size = NULL,
  split_array_by = -1, rettype = "list", fail_on_error = TRUE,
  workers = NULL, log_worker = FALSE, chunk_size = NA,
  timeout = Inf)
```

Arguments

fun	A function to call
...	Objects to be iterated in each function call
const	A list of constant arguments passed to each function call
export	List of objects to be exported to the worker
seed	A seed to set for each function call
memory	Short for template=list(memory=value)
template	A named list of values to fill in template
n_jobs	The number of LSF jobs to submit; upper limit of jobs if job_size is given as well
job_size	The number of function calls per job
split_array_by	The dimension number to split any arrays in '...'; default: last
rettype	Return type of function call (vector type or 'list')
fail_on_error	If an error occurs on the workers, continue or fail?
workers	Optional instance of QSys representing a worker pool
log_worker	Write a log file for each worker
chunk_size	Number of function calls to chunk together defaults to 100 chunks per worker or max. 10 kb per chunk
timeout	Maximum time in seconds to wait for worker (default: Inf)

Value

A list of whatever 'fun' returned

Examples

```
## Not run:
# Run a simple multiplication for numbers 1 to 3 on a worker node
fx = function(x) x * 2
Q(fx, x=1:3, n_jobs=1)
# list(2,4,6)

# Run a mutate() call in dplyr on a worker node
iris %>%
    mutate(area = Q(`*`, e1=Sepal.Length, e2=Sepal.Width, n_jobs=1))
# iris with an additional column 'area'

## End(Not run)
```

QSys *Class for basic queuing system functions*

Description

Provides the basic functions needed to communicate between machines This should abstract most functions of rZMQ so the scheduler implementations can rely on the higher level functionality

Usage

```
QSys
```

Format

An object of class `R6ClassGenerator` of length 24.

Q_rows *Queue function calls defined by rows in a data.frame*

Description

Queue function calls defined by rows in a data.frame

Usage

```
Q_rows(df, fun, const = list(), export = list(), seed = 128965,
  memory = NULL, template = list(), n_jobs = NULL, job_size = NULL,
  rettype = "list", fail_on_error = TRUE, workers = NULL,
  log_worker = FALSE, chunk_size = NA, timeout = Inf)
```

Arguments

df	data.frame with iterated arguments
fun	A function to call
const	A list of constant arguments passed to each function call
export	List of objects to be exported to the worker
seed	A seed to set for each function call
memory	Short for template=list(memory=value)
template	A named list of values to fill in template
n_jobs	The number of LSF jobs to submit; upper limit of jobs if job_size is given as well
job_size	The number of function calls per job
rettype	Return type of function call (vector type or 'list')

fail_on_error	If an error occurs on the workers, continue or fail?
workers	Optional instance of QSys representing a worker pool
log_worker	Write a log file for each worker
chunk_size	Number of function calls to chunk together defaults to 100 chunks per worker or max. 10 kb per chunk
timeout	Maximum time in seconds to wait for worker (default: Inf)

---

register_dopar_cmq *Register clustermq as 'foreach' parallel handler*

Description

Register clustermq as 'foreach' parallel handler

Usage

```
register_dopar_cmq(...)
```

Arguments

...	List of arguments passed to the 'Q' function, e.g. n_jobs

---

SGE *SGE scheduler functions*

Description

Derives from QSys to provide SGE-specific functions

Usage

```
SGE
```

Format

An object of class `R6ClassGenerator` of length 24.

SLURM *SLURM scheduler functions*

Description

Derives from QSys to provide SLURM-specific functions

Usage

```
SLURM
```

Format

An object of class `R6ClassGenerator` of length 24.

SSH *SSH scheduler functions*

Description

Derives from QSys to provide SSH-specific functions

Usage

```
SSH
```

Format

An object of class `R6ClassGenerator` of length 24.

ssh_proxy *SSH proxy for different schedulers*

Description

Do not call this manually, the SSH qsys will do that

Usage

```
ssh_proxy(ctl, job, qsys_id = qsys_default)
```

Arguments

`ctl`	The port to connect to the master for proxy control
`job`	The port to connect to the master for job control
`qsys_id`	Character string of QSys class to use

summarize_result *Print a summary of errors and warnings that occurred during processing*

Description

Print a summary of errors and warnings that occurred during processing

Usage

```
summarize_result(result, n_errors, n_warnings, cond_msgs,
  at = length(result), fail_on_error = TRUE)
```

Arguments

result	A list or vector of the processing result
n_errors	How many errors occurred
n_warnings	How many warnings occurred
cond_msgs	Error and warnings messages, we display first 50
at	How many calls were procesed up to this point
fail_on_error	Stop if error(s) occurred

vec_lookup *Lookup table for return types to vector NAs*

Description

Lookup table for return types to vector NAs

Usage

```
vec_lookup
```

Format

An object of class `list` of length 9.

worker *R worker submitted as cluster job*

Description

Do not call this manually, the master will do that

Usage

```
worker(master, timeout = 600, ..., verbose = TRUE)
```

Arguments

master	The master address (tcp://ip:port)
timeout	Time until worker shuts down without hearing from master
...	Catch-all to not break older template values (ignored)
verbose	Whether to print debug messages

workers *Creates a pool of workers*

Description

Creates a pool of workers

Usage

```
workers(n_jobs, data = NULL, reuse = TRUE, template = list(),
  log_worker = FALSE, qsys_id = getOption("clustermq.scheduler",
  qsys_default))
```

Arguments

n_jobs	Number of jobs to submit (0 implies local processing)
data	Set common data (function, constant args, seed)
reuse	Whether workers are reusable or get shut down after call
template	A named list of values to fill in template
log_worker	Write a log file for each worker
qsys_id	Character string of QSys class to use

Value

An instance of the QSys class

WORKER_DONE *Message ID indicating worker is shutting down*

Description

Field has to be ‘time‘ with a object returned by ‘Sys.time()‘

Usage

```
WORKER_DONE
```

Format

An object of class `integer` of length 1.

WORKER_READY *Message ID indicating worker is accepting jobs*

Description

It may contain the field ‘result‘ with a finished chunk

Usage

```
WORKER_READY
```

Format

An object of class `integer` of length 1.

WORKER_STOP *Message ID telling worker to stop*

Description

No fields

Usage

```
WORKER_STOP
```

Format

An object of class `integer` of length 1.

WORKER_UP *Message ID indicating worker is accepting jobs*

Description

Field has to be ‘worker_id‘ to master or empty to ssh_proxy Answer is serialized common data (‘fun‘, ‘const‘, and ‘seed‘) or ‘redirect‘ (with URL where worker can get data)

Usage

```
WORKER_UP
```

Format

An object of class `integer` of length 1.

work_chunk *Function to process a chunk of calls*

Description

Each chunk comes encapsulated in a data.frame

Usage

```
work_chunk(df, fun, const_args = list(), rettype = "list",
  common_seed = NULL, progress = FALSE)
```

Arguments

`df`	A data.frame with call IDs as rownames and arguments as columns
`fun`	The function to call
`const_args`	Constant arguments passed to each call
`rettype`	Return type of function
`common_seed`	A seed offset common to all function calls
`progress`	Logical indicated whether to display a progress bar

Value

A list of call results (or try-error if they failed)

Index