Debugging foreach and doRedis programs

Debugging parallel programs is hard. R's foreach and doRedis packages include a few options outlined in this document to help.

Error handling

The `.errorhandling` parameter of the `foreach()` function takes a character value among "stop", "remove", or "pass", defaulting to "stop." That means that as soon as an error is detected in any loop iteration, the entire foreach loop stops with that error.

Setting `.errorhandling = "pass"` is very useful for debugging your programs since all errors encountered in the loop are returned verbatim in your solution. Start experimenting with this option without using the `.combine` parameter to return all results in a list. Consider the following example program that is rigged to explicitly return an error in one of the loop iterations. We use the 'doSEQ' back end below, but this example works exactly the same using any foreach back end.

```
library(foreach)
registerDoSEQ()

foreach(j=1:3, .errorhandling="pass") %dopar%
{
  if(j==2) j + undefined_variable   else j
}

# [[1]]
# [1] 1
#
# [[2]]
# <simpleError in eval(expr, envir, enclos):
#          object 'undefined_variable' not found>
#
# [[3]]
# [1] 3
```

The example shows the error when `j=2`.

doRedis logging

Use the `log` and `loglevel` options in the doRedis `redisWorker()` and `startLocalWorkers()` functions to direct output to a file. Use the `logger()` function inside your foreach loop to write messages to the log. The logger function appends process ID and time stamps to your message. Alternatively

use the plain old R `message()` function instead. These options together provide a kind of "printf"-style debug facility–crude but sometimes effective. Here is an example:

```
library(doRedis)
registerDoRedis("cazart")

startLocalWorkers(n=1, timeout=1, queue="cazart",
                  log="/tmp/cazart.log", loglevel=1)

x <- foreach(j=1:3) %dopar%
{
  logger(paste("hello from loop iteration ", j))
  j
}

# Remove the work queue (terminating the worker process)
removeQueue("cazart")
Sys.sleep(2) # wait for the worker to terminate

# Display the log file
file.show("/tmp/cazart.log")

# Processing task(s) 1...1 from queue cazart ID 9ba32925bee...
# @ 2016-04-23 20:17:26 hostname 3406   hello from loop iteration  1
# Processing task(s) 2...2 from queue cazart ID 9ba32925bee...
# @ 2016-04-23 20:17:26 hostname 3406   hello from loop iteration  2
# Processing task(s) 3...3 from queue cazart ID 9ba32925bee...
# @ 2016-04-23 20:17:26 hostname 3406   hello from loop iteration  3
# Normal worker exit.
```

Note that log files from workers running on different hosts are stored on the corresponding host file systems.

doRedis job and task accounting

The doRedis package includes `setProgress()`, `jobs()` and `tasks()` functions that list detailed information about all processes working for doRedis.

The function

```
setProgress(TRUE)
```

enables an R progress meter during foreach execution. While a foreach loop is running, open a separate R terminal session and use the `jobs()` and `tasks()` functions to list details about running operations. See the help pages for those functions for complete details.

Interactive debugging

A very low-level but effective debugging approach works as follows:

1. Start a single R worker process running in an R terminal using the `redisWorker()` function.
2. Start, in a separate terminal, a master R process running a foreach job.

You can monitor the worker R process as the work progresses in its terminal, even including interactive function debugging. With the `loglevel=1` argument, loop task information is printed to standard output in the R worker process terminal window as it occurs, as well as any message output from the `logger()` function.

Here is an example split into two R sessions, one for the worker and one for the master (it's assumed that you run these in separate R terminal windows). All standard R debugging options are available in each R session, for example use `option(error=recover)` to explore the call stack after encountering an error condition.

```
library(doRedis)
registerDoRedis("cazart")
x <- foreach(j=1:3) %dopar%
{
  logger(paste("  log message from iteration", j))
  j
}
removeQueue("cazart")  # remove the work queue
```

```
library(doRedis)
redisWorker(queue="cazart", loglevel=1, timeout=1)

# Waiting for doRedis jobs.
# Processing task(s) 1...1 from queue cazart ID 9ba77ba191c...
# @ 2016-04-23 21:09:04 homer 3882   log message from iteration 1
# Processing task(s) 2...2 from queue cazart ID 9ba77ba191c...
# @ 2016-04-23 21:09:04 homer 3882   log message from iteration 2
# Processing task(s) 3...3 from queue cazart ID 9ba77ba191c...
# @ 2016-04-23 21:09:04 homer 3882   log message from iteration 3
# Normal worker exit.
```