6. Distributed File System for Subsets of Objects

Chuanhai Liu

Department of Statistics, Purdue University

2016

Table of Contents

6.1 More SupR additions

Everything is an object. Loaded objects have a memory address. SupR objects can alternatively be accessed by their memory address.

It is not recommended for ordinary computing in SupR, but is provided in case it can be helpful in debugging interactive multithreaded programming, where dead locks can occur and debugging can be challenging.

address displays the memory address in the hexadecimal format as a character. Different variables can point to the same objects. Try the following example:

```
> a <- 1:10
> b <- a
> address(a)
0x28a4030
> address(b)
0x28a4030
```

What do you see in your R session?

6.1 More SupR additions

address symbol In the current version of SupR, symbols formed from memory address characters can be used to access the R object in the exact same location.

```
> `0x28a4030`
[1] 1 2 3 4 5 6 7 8 9
> address(`0x28a4030`)
0x28a4030
```

6.2 R namespace: an overview

An overview can be found in the second set of slides. Taking a look at the following topics, we should see a set of functions for understanding what are available to R users on R namespaces.

- `?getNamespace`
- `?loadNamespace`

In particular, the `loadedNamespaces()` call tells us that R name spaces are a collection (internally as an environment object) of loaded namespaces.

```
> loadedNamespaces()
[1] "base" "datasets" ...
```

6.3 Distributed files as databases

A file system has a tree structure. Everyone should be familiar with this. Otherwise, you can do the following simple exercise:

- Type the `pwd` command to see your present working directory in the file system of your unix-like computer.
- Type the `cd /` command to jump to the top-level or root directory of your file system.
- Type the `ls` command to see what subdirectories (and files) are there.
- Use the `cd` command go step by step to your home directory.
- You can also use the `cd` command go to any directory with its so-called absolute path.

6.3 Distributed files as databases

What do we know?

- For linked objects having a tree-type structure, such as file systems, we can represent them with our fimiliar environment objects.
- All file systems that we use in SupR can be considered as a collection of environments.

Here is what we have (or will have) in parallel with R namespaces:

- `connectedDatabases()` shows all the connected file systems, which we call databases.
- `connectDatabase(name, ...)` connects or attaches the database with the given name.
- `disconnectDatabase(name)` disconnects or detaches the named database.
- `getDatabaseEnv(name)` returns the environment object of the connected database specified by the `name` argument.

6.4 Environment objects as databases

Tree nodes can be represented by envir objects and tree leaves can be put in envir objects.

Once represented as environment objects, organized objects can be accessed by the usual way of stepping through environments with the '$' operator.

```
> getDatabase("cluster")$lr
...
```

SupR also allows you to use a somewhat simplified notation, adopting the internet convention for **url**s:

name://toplevel/secondlevel/...

which is equivalent to

getDatabaseEnv(name)$toplevel$secondlevel$...

When `name:` is omitted, the built-in `cluster:` is used as the default database.

6.5 Cluster distributed objects

SupR comes with a built-in distributed file system (DFS). The built-in DFS server is managed by a set of what are called block managers.

- These block managers are started when SupR runs in the cluster mode.
- To be run in the cluster mode, a cluster master, set of workers, and one or more drivers are working together, creating the cluster computing environment for the user.
- In the cluster mode, you can access distributed objects with functions such as `ls`, without having to think about the exact locations the distributed objects are.

More discussions are to be given in the slides on cluster computing.

The current experimental `dfs.*` functions will be fully integrated, hopefully, with what is described here in later versions.

6.6 The put and assign functions

The current `dfs.put` and `distribute` functions can/may be made as special cases of `put` and `assign` functions.

The usual assignment operator `<-` should also be made to work with distributed objects.

6.7 The get function

The current `dfs.get` function can/may be made as a special case of `get` function on environment variables.

The usual accessing objects by name should also be made to work with distributed objects.

6.8 The ls and objects functions

The current `dfs.ls` functions can/may be made as a special case of `ls` function on environment variables.

6.9 The rm function

The current `dfs.rm` functions can/may be made as a special case of `rm` function on environment variables.

6.10 Attach external distributed file systems and databases

The famous HDFS has already been made to work with the current experimental `dfs.attach` function.

This may be integrated or modified to use the `connectDatabase()` function.

With more development, SupR should be able to work with external databases without much challenges.

6.11 Exercises