Accessing Redis Data Caches via Rcpp

Dr Dirk Eddelbuettel
dirk@eddelbuettel.com
@eddelbuettel

Chicago R Users Group
Chicago, IL
14 May 2014

Outline

Overview

Why the hype?

- **Simple**: Does one thing, and does it well
- **Fast**: Run `redis-benchmark` to see just how fast
- **Widely used**: Twitter, GitHub, Craigslist, StackOverflow, . . .
- **Multi-language**: Bindings from anything you may use
- **Active**: Well maintained and documented

Write from Python

```
#!/usr/bin/python

import redis

redishost   = "localhost"
redisserver = redis.StrictRedis(redishost)

key = "ex:ascii:simpleString"
val = "abracadabra"
res = redisserver.set(key, val)
```

Read in R

```r
library(rredis)

redisConnect()

key <- "ex:ascii:simpleString"
val <- redisGet(key)
cat("Got", val, "from", key, "\n")

## Got abracadabra from ex:ascii:simpleString
```

Or read in Shell

```
$ redis-cli get ex:ascii:simpleString
"abracadabra"
$
```

More generally

We can

- Read
- Write

from just about any programming language or shell.

(So far) all we require is string processing.

Data Structures

Redis supports many relevant data types:

- Strings
- Hashes
- Lists
- Sets
- Sorted Sets

as well as transactions, key management, pub/sub, embedded scripting, connection management and more.

rredis

Wonderful package by Bryan Lewis that covers (all of ?) Redis

Awesome for things like
`redisSet("myModel", lm(someFormula, someData))`

(Mostly) efficient enough.

Uses string format exclusively.

Automagically deploys R serialization.

Also used as backend for **doRedis**

Simple helper functions

```
redisConnect("someServer.some.net")

rput <- function(X) {
    xstr <- deparse(substitute(X))
    redisSet(xstr, X)
}

rget <- function(key) {
    val <- redisGet(key)    # default instance
    redisDelete(key)
    invisible(val)
}
```

Even nicer: memoise by Michael Kane

```r
require(rredis)
redisConnect()

memoize <- function(expr, key=NULL, expire_time=Inf,
                    verbose=FALSE, envir=parent.frame()) {
  if (is.null(key)) {
    key <- paste(substitute(expr), collapse="")
  }
  if (redisExists(key)) {
    ret <- redisGet(key)
  } else {
    ret <- eval(substitute(expr), envir=envir)
    redisSet(key, ret)
  }
  if (expire_time < Inf) {
    redisExpireAt(proj_doc_key,
      as.integer(as.POSIXct(Sys.time())+expire_time))
  }
  ret
}
```

Outline

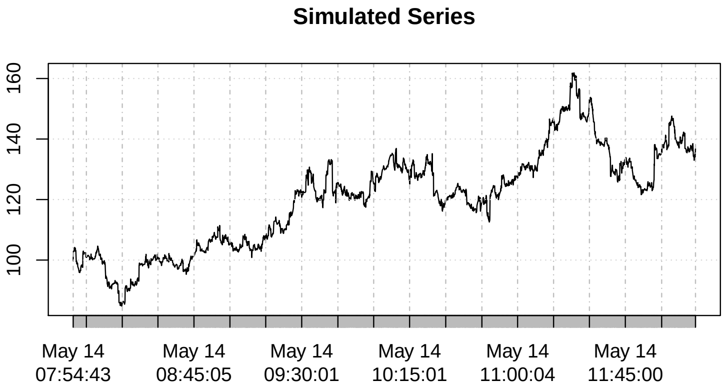
Time series

Our basic premise and idea is to deploy disconnected writers (middleware clients in C, C++, Python, ...) and consumers (R) – by placing Redis in the middle.

But for “longer” time series the combined cost of deserialization and parsing is too high in R.

Example

```
set.seed(123); N <- 2500
x <- xts(100*cumprod(1+rnorm(N)*0.005 +
              (runif(N)>0.95)*rnorm(N)*0.025),
         order.by=Sys.time()+cumsum(exp(3*runif(N))))
plot(x, main="Simulated Series", type='l')
```



Writing and Reading

With **rredis** we set and get the time series as follows:

```
setAsAscii <- function(dat) {
    N <- nrow(dat)
    ## insertion is row by row
    for (i in 1:N) {
        redisZAdd("ex:ascii:series",
                  dat[i,1], dat[i,])
    }
}

## retrieval is by list
getFromAscii <- function() {
    xx <- do.call(rbind,
                  redisZRange("ex:ascii:series", 0, -1))
    xt <- xts(xx[,-1],
              order.by=as.POSIXct(xx[,1], origin="1970-01-01"))
}
```

RApiSerialize

A (fairly new) CRAN package we released recently.

It does just one thing: give us serialization and deserialization from the R API at the C(++) level.

It is used by **RcppRedis**, and provides it with C-level (de-)serialization without having to call “up” to R.

RcppRedis

A (fairly new) (and higly incomplete) CRAN package (as of yesterday).

It covers just a couple of commands, but those run rather fast.

Writing and Reading

```
setAsBinary <- function(dat) {
    redis$zadd("ex:bin:series", as.matrix(dat))
}

getFromBinary <- function() {
    zz <- redis$zrange("ex:bin:series", 0, -1)
    zt <- xts(zz[,-1],
              order.by=as.POSIXct(zz[,1], origin="1970-01-01"))
}
```

Writing and Reading – Part Two

```
// redis "zadd" -- insert score + matrix row (no R serial.)
// by convention, 1st elem of row vector is the score value
double zadd(std::string key, Rcpp::NumericMatrix x) {
  double res = 0;
  for (int i=0; i<x.nrow(); i++) {
     Rcpp::NumericVector y = x.row(i);
     // uses binary protocol, see hiredis doc at github
     redisReply *reply =
       static_cast<redisReply*>(redisCommand(prc_,
                                             "ZADD %s %f %b",
                                             key.c_str(),
                                             y[0],
                                             y.begin(),
                                             y.size()*szdb));
    checkReplyType(reply, replyInteger_t);
    res += static_cast<double>(reply->integer);
    freeReplyObject(reply);
  }
  return(res);
}
```

Net Effect: demo/simDemo.R

```
##
## Writing
##
             test replications elapsed relative
setAsBinary(dat)            1   0.127    1.000
 setAsAscii(dat)            1 100.001  787.409
##
## Reading
##
            test replications elapsed relative
getFromBinary()           10   0.031    1.000
 getFromAscii()           10   4.792  154.581
```

RcppRedis Open Questions

Right now the **RcppRedis** package straddles three worlds:

- Strings to communicate with Python, C++, cmdline, ...
- Raw R strings and (de-)serialization to talk to **rredis**
- Binary data (as vectors) for efficient time series storage.

We don't plan to provide the cross-product of encodings and commands, but rather pick and choose.

We now have **Shiny** apps that slice and dice (near) real-time series related to trading. And I am not going to say more.

Redis and Rcpp Summary

This short talk tried to convince you that

- Redis is cooler than sliced bread.
- **rredis** is a wonderful package you should use.
- Redis also allows binary connection.
- (Lots of) string-to-numeric conversions are slow.
- **Rcpp** is ready, willing and able to help.
- **RcppRedis** helps overcome a few bottlenecks.

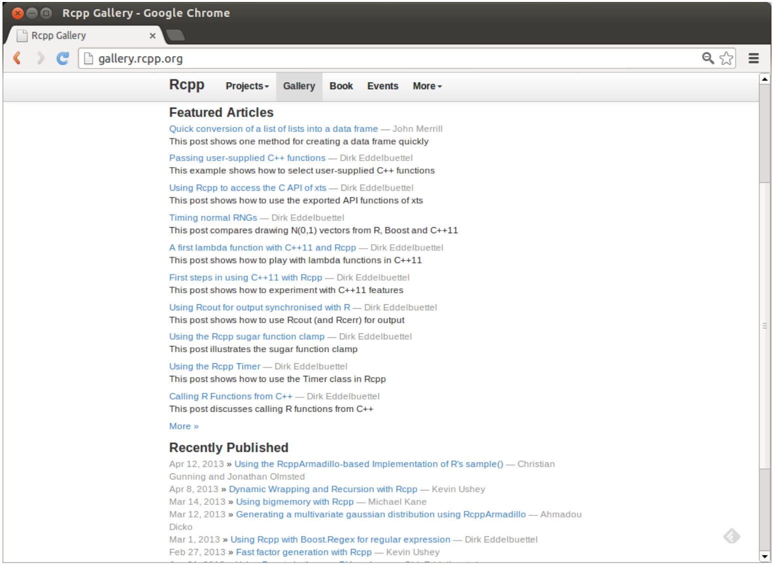
RcppRedis is open for collaboration. See what it does, see what it misses, and consider contributing to it.

Outline

Documentation

- The **Rcpp** package comes with **eight pdf vignettes**, and numerous help pages.
- The introductory vignettes are now **published** (Rcpp and RcppEigen in *J Stat Software*, RcppArmadillo in *Comp. Stat.& Data Anal.*).
- The **rcpp-devel** list is *the* recommended resource, generally very helpful, and fairly low volume.
- **StackOverflow** now has almost 500 posts too.
- And a number of **blog posts** introduce/discuss features.
- Plus . . .

Rcpp Gallery



Featured Articles

Quick conversion of a list of lists into a data frame — John Merrill
This post shows one method for creating a data frame quickly

Passing user-supplied C++ functions — Dirk Eddelbuettel
This example shows how to select user-supplied C++ functions

Using Rcpp to access the C API of xts — Dirk Eddelbuettel
This post shows how to use the exported API functions of xts

Timing normal RNGs — Dirk Eddelbuettel
This post compares drawing N(0,1) vectors from R, Boost and C++11

A first lambda function with C++11 and Rcpp — Dirk Eddelbuettel
This post shows how to play with lambda functions in C++11

First steps in using C++11 with Rcpp — Dirk Eddelbuettel
This post shows how to experiment with C++11 features

Using Rcout for output synchronised with R — Dirk Eddelbuettel
This post shows how to use Rcout (and Rcerr) for output

Using the Rcpp sugar function clamp — Dirk Eddelbuettel
This post illustrates the sugar function clamp

Using the Rcpp Timer — Dirk Eddelbuettel
This post shows how to use the Timer class in Rcpp

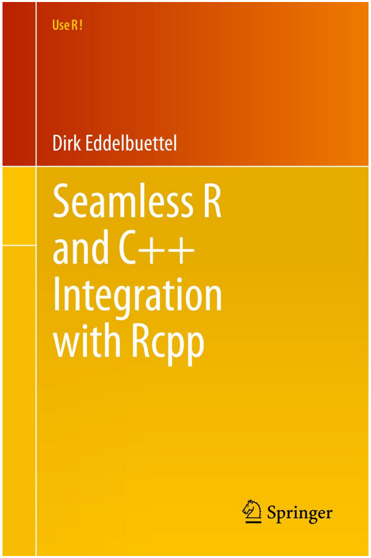
Calling R Functions from C++ — Dirk Eddelbuettel
This post discusses calling R functions from C++

More »

Recently Published

Apr 12, 2013 » Using the RcppArmadillo-based Implementation of R's sample() — Christian Gunning and Jonathan Olmsted
Apr 8, 2013 » Dynamic Wrapping and Recursion with Rcpp — Kevin Ushey
Mar 14, 2013 » Using bigmemory with Rcpp — Michael Kane
Mar 12, 2013 » Generating a multivariate gaussian distribution using RcppArmadillo — Ahmadou Dicko
Mar 1, 2013 » Using Rcpp with Boost.Regex for regular expression — Dirk Eddelbuettel
Feb 27, 2013 » Fast factor generation with Rcpp — Kevin Ushey

The Rcpp book



Use R!

Dirk Eddelbuettel

Seamless R and C++ Integration with Rcpp

Springer

In print since June 2013