On the lambda (http://www.onthelambda.co

another blog from a data analyst – by tony fischetti

Squeezing more speed from R for nothing, Rcpp style

[email protected] (http://www.onthelambda.com/author/tony-fischettigmail-com/) June 27, 2014 (http://www.onthelambda.com/2014/06/27/squeezing-more-speed-from-r-for-nothing-rcpp-style/) 11 Comments (http://www.onthelambda.com/2014/06/27/squeezing-more-speed-from-r-for-nothing-rcpp-style/#comments)

In a previous post (http://www.onthelambda.com/2013/11/13/parallel-r-and-air-travel/) we explored how you can greatly speed up certain types of long-running computations in R by parallelizing your code using multicore package*. I also mentioned that there were a few other ways to speed up R code; the one I will be exploring in this post is using Rcpp to replace time-critical inner-loops with C++.

In general, good C++ code almost always runs faster than equivalent R code. Higher level language affordances like garbage collection, dynamic typing, and bounds checking can add a lot of computational overhead. Further, C/C++ compiles down to machine code, whereas R byte-code has to be interpreted.

On the other hand, I would hate to do all my statistics programming in a language like C++, precisely *because* of those higher-level language affordances I mentioned above. When development time (as opposed to execution time) is taken into account, programming in R is much faster for me (and makes me a very happy programmer).

On occasion, though, there are certain sections of R code that I wish I could rewrite in C/C++. They may be simple computations that get called thousands or millions of times in a loop. If I could just write these time-critical snippets with C/C++ and not have to throw the proverbial baby out with the bath water (and rewrite everything in C), my code would run much faster.

Though there have been packages to make this sort of thing since the early 2000s, Rcpp (and the Rcpp family**) has made this even easier; now interfacing with R objects is seamless.

To show an example of how you might use Rcpp, I've used the same example from my post "Parallel R (and air travel)" (http://www.onthelambda.com/2013/11/13/parallel-r-and-air-travel/). In this example, we use longitude and latitude info from all US airports to derive the average (mean) distance between every two US airports. The function I will be replacing with C++ is the function to compute the distance between two longitude latitude pairs (Haversine's formula (http://en.wikipedia.org/wiki/Haversine_formula)) on a sphere (which is just an approximation).

The R functions to do this look like this:

```r
to.radians<-function(degrees){
  degrees * pi / 180
}

haversine <- function(lat1, long1, lat2, long2, unit="km"){
  radius <- 6378      # radius of Earth in kilometers
  delta.phi <- to.radians(lat2 - lat1)
  delta.lambda <- to.radians(long2 - long1)
  phi1 <- to.radians(lat1)
  phi2 <- to.radians(lat2)
  term1 <- sin(delta.phi/2) ^ 2
  term2 <- cos(phi1) * cos(phi2) * sin(delta.lambda/2) ^ 2
  the.terms <- term1 + term2
  delta.sigma <- 2 * atan2(sqrt(the.terms), sqrt(1-the.terms))
  distance <- radius * delta.sigma
  if(unit=="km") return(distance)
  if(unit=="miles") return(0.621371*distance)
}
```

While the C++ functions look like this:

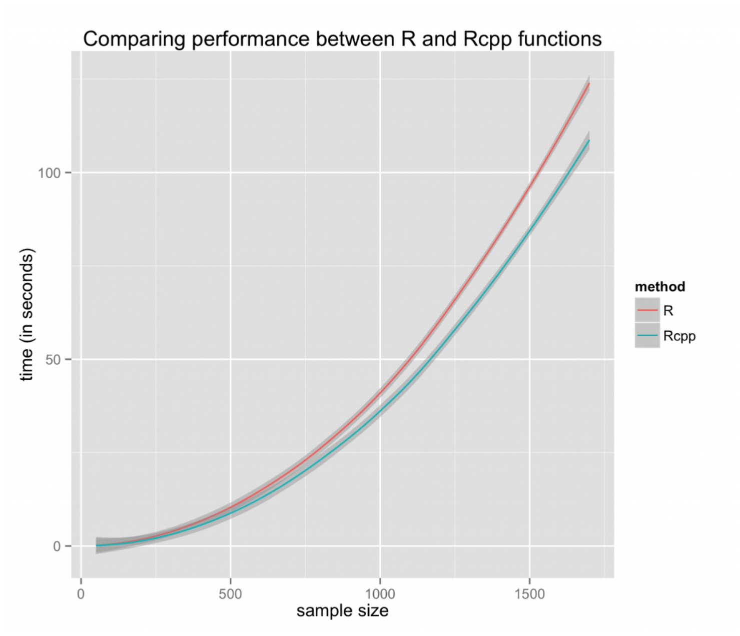
```cpp
#include <iostream>
#include <math.h>
#include <Rcpp.h>

// [[Rcpp::export]]
double to_radians_cpp(double degrees){
    return(degrees * 3.141593 / 180);
}

// [[Rcpp::export]]
double haversine_cpp(double lat1, double long1,
                     double lat2, double long2,
                     std::string unit="km"){
    int radius = 6378;
    double delta_phi = to_radians_cpp(lat2 - lat1);
    double delta_lambda = to_radians_cpp(long2 - long1);
    double phi1 = to_radians_cpp(lat1);
    double phi2 = to_radians_cpp(lat2);
    double term1 = pow(sin(delta_phi / 2), 2);
    double term2 = cos(phi1) * cos(phi2) * pow(sin(delta_lambda/2), 2);
    double the_terms = term1 + term2;
    double delta_sigma = 2 * atan2(sqrt(the_terms), sqrt(1-the_terms));
    double distance = radius * delta_sigma;

    /* if it is anything *but* km it is miles */
    if(unit != "km"){
        return(distance*0.621371);
    }

    return(distance);
}
```

Besides for the semicolons, other assignment operator and the type declarations, these codes are almost identical.

Next, we put the C++ code above in a C++ source file. We will call it, and automatically compile and link to it from our driver R code thusly***:

```r
calc.distance.two.rows <- function(ind1, ind2,
                                   version=haversine){
  return(version(air.locs[ind1, 2],
                 air.locs[ind1, 3],
                 air.locs[ind2, 2],
                 air.locs[ind2, 3]))
}

air.locs <- read.csv("airportcodes.csv", stringsAsFactors=FALSE)


combos <- combn(1:nrow(air.locs), 2, simplify=FALSE)
num.of.comps <- length(combos)


mult.core <- function(version=haversine_cpp){
  the.sum <- sum(unlist(mclapply(combos,
                                 function(x){
                                   calc.distance.two.rows(x[1], x[2],
                                                          version)
                                 },
                                 mc.cores=4)))
  result <- the.sum / num.of.comps
  return(result)
}

mult.core(version=haversine_cpp)
```

Comparing the R version against the C++ version over a range of sample sizes yielded a chart like this:
(http://www.onthelambda.com/wp-content/uploads/2014/06/RvRcppdifferential.png)



To run this to completion would have taken 4 hours but, if my math is correct, rewriting the distance function shaved of over 15 minutes from the completion time.

It is not uncommon for the Rcpp to speed up R code by *orders of magnitude*. In this link, Dirk Eddelbuettel demonstrates an 80-fold speed increase (http://dirk.eddelbuettel.com/blog/2010/09/07/) (albeit with a contrived example).

So why did we not get an 80-fold increase?

I could have (and will) rewrite more of this program in Rcpp to avoid some of the overhead with repeated calls to compiled C++. My point here was more to show that we can use Rcpp to speed up this program with *very little* work–almost for nothing. Again, besides for certain syntactical differences and type declarations, the R and C++ functions are virtually the same.

As you become more comfortable with it–and use it more within the same scripts–Rcpp will likely pay higher and higher dividends.

The next time we revisit this contrived airport example, we will be profiling it expanding the C++ and eventually, use distributed computing to get it as fast as we can.

* the 'multicore' package is now deprecated in favor of 'parallel'

** RCpp11 (for modern C++), RccpEigen (for use of the Eigen C++ linear algebra template library), RCppArmadillo (for use of the Eigen C++ linear algebra template library), and a few others

*** this code is a little bit different than the code in the first airport distance blog post because I switched from using the 'multicore' package to the 'parallel' package

share this:

R (http://www.onthelambda.com/category/r/) high performance computing (http://www.onthelambda.com/tag/high-performance-computing/), R (http://www.onthelambda.com/tag/r/) Bookmark (http://www.onthelambda.com/2014/06/27/squeezing-more-speed-from-r-for-nothing-rcpp-style/)

PREVIOUS ARTICLE
Damn the torpedoes, full speed ahead: making the switch to Python 3 (http://www.onthelambda.com/2014/05/13/damn-the-torpedoes-full-speed-ahead-making-the-switch-to-python-3/)

NEXT ARTICLE
Interactive visualization of non-linear logistic regression decision boundaries with Shiny (http://www.onthelambda.com/2014/07/24/interactive-visualization-of-non-linear-logistic-regression-decision-boundaries-with-shiny/)

11 RESPONSES

OWE JESSEN (HTTP://WWW.ECONINFO.DE/2014/06/19/COMPARING-IFELSE-WITH-C-FOR-LOOP-WITH-IFS/)
June 28, 2014 / 2:52 am

In the link there is a less contrieved example - using Rcpp to replace a nested ifelse command shows 60x speedup.

Reply (http://www.onthelambda.com/2014/06/27/squeezing-more-speed-from-r-for-nothing-rcpp-style/?replytocom=3934#respond)

TONY.FISCHETTI@GMAIL.COM
June 30, 2014 / 11:17 am

That's incredible–it was such a simple change. I'm definitely going to further explore writing vectorized Rcpp functions. Thanks for posting this!

↩ Reply (http://www.onthelambda.com/2014/06/27/squeezing-more-speed-from-r-for-nothing-rcpp-style/?replytocom=3954#respond)

YI TANG
August 18, 2014 / 8:35 am

data transferring between R and C++ takes time, in your case, no less then the time in calculating distance. This is the main reason Rcpp version isn't xx times faster.

↩ Reply (http://www.onthelambda.com/2014/06/27/squeezing-more-speed-from-r-for-nothing-rcpp-style/?replytocom=4797#respond)

TONY.FISCHETTI@GMAIL.COM
August 19, 2014 / 9:51 am

Thanks, that's a great point! I'll be refactoring this in a latter post.

↩ Reply (http://www.onthelambda.com/2014/06/27/squeezing-more-speed-from-r-for-nothing-rcpp-style/?replytocom=4820#respond)

Pingback: Lessons learned in high-performance R – On the lambda (http://www.onthelambda.com/2015/05/31/lessons-learned-in-high-performance-r/)

JAMES CURRAN (HTTP://WWW.JAMESCURRAN.CO.NZ)
July 13, 2015 / 3:51 am

Tony,

I know this isn't the point, but I bet you could also achieve a bit more speed by replacing many of the divisions with multiplication, e.g. dividing by two costs way more than multiplying by 0.5. Also this one : degrees * 3.141593 / 180 - way faster to multiply by 0.01745329 and also, not make a function call which has its own overhead.

James

↩ Reply (http://www.onthelambda.com/2014/06/27/squeezing-more-speed-from-r-for-nothing-rcpp-style/?replytocom=9261#respond)

TONY.FISCHETTI@GMAIL.COM
August 2, 2015 / 5:15 pm

Wow, I had no idea. That's great to know–thanks!

↩ Reply (http://www.onthelambda.com/2014/06/27/squeezing-more-speed-from-r-for-nothing-rcpp-style/?replytocom=9486#respond)

Pingback: fReigeist (http://freigeist.devmag.net/economics/936-936.html)

Pingback: Using R To Estimate Spatial HAC Errors Per Conley | Thiemo René Fetzer (http://www.trfetzer.com/using-r-to-estimate-spatial-hac-errors-per-conley/)

Pingback: Correction For Spatial And Temporal Auto-Correlation In Panel Data: Using R To Estimate Spatial HAC Errors Per Conley | Thiemo René Fetzer (http://www.trfetzer.com/correction-for-spatial-and-temporal-auto-correlation-in-panel-data-using-r-to-estimate-spatial-hac-errors-per-conley/)

Pingback: Correction For Spatial And Temporal Auto-Correlation In Panel Data: Using R To Estimate Spatial HAC Errors Per Conley | Mubashir Qasim (http://mqasim.me/?p=37422)

Leave a Reply

Your email address will not be published. Required fields are marked *

Message...

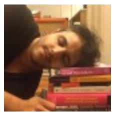
Name*

Email*

Website

POST COMMENT

ABOUT ME


Data scientist. Vegetarian. Interests lie in R, python, art, cognition, game theory, machine learning, books, ethics, post-punk, human/animal rights, and oxford commas.

Check out my book, Data Analysis with R (http://bit.ly/data-analysis-with-r)

Follow @tonyfischetti

(https://www.facebook.com/tony.fischetti) (https://github.com/tonyfischetti) (https://twitter.com/tonyfischetti)

TAG CLOUD

awk (http://www.onthelambda.com/tag/awk/) classification (http://www.onthelambda.com/tag/classification/) D3 (http://www.onthelambda.com/tag/d3/) data mining (http://www.onthelambda.com/tag/data-mining/) datavis (http://www.onthelambda.com/tag/datavis/) debian (http://www.onthelambda.com/tag/debian/) dplyr (http://www.onthelambda.com/tag/dplyr/) evolutionary biology (http://www.onthelambda.com/tag/evolutionary-biology/) functional programming (http://www.onthelambda.com/tag/functional-programming/) ggplot2 (http://www.onthelambda.com/tag/ggplot2/) graph theory (http://www.onthelambda.com/tag/graph-theory/) haskell (http://www.onthelambda.com/tag/haskell/) high performance computing (http://www.onthelambda.com/tag/high-performance-computing/) hypothesis testing (http://www.onthelambda.com/tag/hypothesis-testing/) linguistics (http://www.onthelambda.com/tag/linguistics/) lisp (http://www.onthelambda.com/tag/lisp/) mac (http://www.onthelambda.com/tag/mac/) machine learning (http://www.onthelambda.com/tag/machine-learning/) magrittr (http://www.onthelambda.com/tag/magrittr/) markov chains (http://www.onthelambda.com/tag/markov-chains/) monads (http://www.onthelambda.com/tag/monads/) music (http://www.onthelambda.com/tag/music/) natural language processing (http://www.onthelambda.com/tag/natural-language-processing/) nhst (http://www.onthelambda.com/tag/nhst/) open data (http://www.onthelambda.com/tag/open-data/) optical character recognition (http://www.onthelambda.com/tag/optical-character-recognition/) os x (http://www.onthelambda.com/tag/os-x/) p-values (http://www.onthelambda.com/tag/p-values/) pedagogy (http://www.onthelambda.com/tag/pedagogy/) perl (http://www.onthelambda.com/tag/perl/) python (http://www.onthelambda.com/tag/python/) R (http://www.onthelambda.com/tag/r/) reading (http://www.onthelambda.com/tag/reading/) recommender systems (http://www.onthelambda.com/tag/recommender-systems/) research (http://www.onthelambda.com/tag/research/) sake (http://www.onthelambda.com/tag/sake/) science (http://www.onthelambda.com/tag/science/) scientific workflow systems (http://www.onthelambda.com/tag/scientific-workflow-systems/) self-tracking (http://www.onthelambda.com/tag/self-tracking/) shell (http://www.onthelambda.com/tag/shell/) shiny (http://www.onthelambda.com/tag/shiny/) software (http://www.onthelambda.com/tag/software/) statistics (http://www.onthelambda.com/tag/statistics/) twitter (http://www.onthelambda.com/tag/twitter/) unix (http://www.onthelambda.com/tag/unix/)

RECENT POSTS

- Data validation with the assertr package (http://www.onthelambda.com/2017/03/20/data-validation-with-the-assertr-package/)
- The Bayesian approach to ridge regression (http://www.onthelambda.com/2016/10/30/the-bayesian-approach-to-ridge-regression/)
- Using Python decorators to be a lazy programmer: a case study (http://www.onthelambda.com/2016/07/08/using-python-decorators-to-be-a-lazy-programmer-a-case-study/)
- Computational foreign language learning: a study in Spanish verbs usage (http://www.onthelambda.com/2016/06/30/computational-foreign-language-learning-a-study-in-spanish-verbs-usage/)
- Genre-based Music Recommendations Using Open Data (and the problem with recommender systems) (http://www.onthelambda.com/2016/01/11/genre-based-music-recommendations-using-open-data-and-the-problem-with-recommender-systems/)
- Kickin' it with elastic net regression (http://www.onthelambda.com/2015/08/19/kickin-it-with-elastic-net-regression/)
- Lessons learned in high-performance R (http://www.onthelambda.com/2015/05/31/lessons-learned-in-high-performance-r/)
- The hardest thing about teaching statistics (http://www.onthelambda.com/2015/04/30/the-hardest-thing-about-teaching-statistics/)
- I'm all about that bootstrap ('bout that bootstrap) (http://www.onthelambda.com/2015/03/21/im-all-about-that-bootstrap-bout-that-bootstrap/)
- Playing around with #rstats twitter data (http://www.onthelambda.com/2015/02/28/playing-around-with-rstats-twitter-data/)
- Assertive R programming in dplyr/magrittr pipelines (http://www.onthelambda.com/2015/01/23/assertive-r-programming-in-dplyrmagrittr-pipelines/)
- What does Flatland have to do with Haskell? (http://www.onthelambda.com/2014/12/22/what-does-flatland-have-to-do-with-haskell/)
- Sending text messages at random times using python (http://www.onthelambda.com/2014/11/15/sending-text-messages-at-random-times-using-python/)
- Why is my OS X Yosemite install taking so long?: an analysis (http://www.onthelambda.com/2014/10/23/why-is-my-os-x-yosemite-install-taking-so-long-an-analysis/)
- Fun with .Rprofile and customizing R startup (http://www.onthelambda.com/2014/09/17/fun-with-rprofile-and-customizing-r-startup/)
- Interactive visualization of non-linear logistic regression decision boundaries with Shiny (http://www.onthelambda.com/2014/07/24/interactive-visualization-of-non-linear-logistic-regression-decision-boundaries-with-shiny/)
- Squeezing more speed from R for nothing, Rcpp style (http://www.onthelambda.com/2014/06/27/squeezing-more-speed-from-r-for-nothing-rcpp-style/)
- Damn the torpedoes, full speed ahead: making the switch to Python 3 (http://www.onthelambda.com/2014/05/13/damn-the-torpedoes-full-speed-ahead-making-the-switch-to-python-3/)
- Take a look, it's in a book: distribution of kindle e-book highlights (http://www.onthelambda.com/2014/04/10/take-a-look-its-in-a-book-distribution-of-kindle-e-book-highlights/)

- How to make an absurd twitter bot in python (http://www.onthelambda.com/2014/03/20/how-to-make-an-absurd-twitter-bot-in-python/)
- How to fake a sophisticated knowledge of wine with Markov Chains (http://www.onthelambda.com/2014/02/20/how-to-fake-a-sophisticated-knowledge-of-wine-with-markov-chains/)
- How dplyr replaced my most common R idioms (http://www.onthelambda.com/2014/02/10/how-dplyr-replaced-my-most-common-r-idioms/)
- Using Last.fm to data mine my music listening history (http://www.onthelambda.com/2014/01/27/using-last-fm-to-data-mine-my-music-listening-history/)
- The performance gains from switching R's linear algebra libraries (http://www.onthelambda.com/2013/12/26/the-performance-gains-from-switching-rs-linear-algebra-libraries/)
- Visualizing data analysis pipelines using NetworkX (http://www.onthelambda.com/2013/12/08/visualizing-data-analysis-pipelines-using-networkx/)
- Compiling R from source and why you shouldn't do it (http://www.onthelambda.com/2013/11/22/compiling-r-from-source-and-why-you-shouldnt-do-it/)
- Parallel R (and air travel) (http://www.onthelambda.com/2013/11/13/parallel-r-and-air-travel/)
- qstats - quick and dirty statistics tool for the Unix pipeline (http://www.onthelambda.com/2013/11/05/qstats-quick-and-dirty-statistics-tool-for-the-unix-pipeline/)
- Linguistics, meet Evolutionary Biology (http://www.onthelambda.com/2013/10/29/linguistics-meet-evolutionary-biology/)
- Unsupervised correction of optical character misrecognition (http://www.onthelambda.com/2013/10/22/unsupervised-correction-of-optical-character-misrecognition/)
- The state of package management on Mac OS X (http://www.onthelambda.com/2013/10/14/the-state-of-package-management-on-mac-os-x/)
- On the treachery of point-and-click "black-box" data analysis (http://www.onthelambda.com/2013/10/07/on-the-treachery-of-point-and-click-black-box-data-analysis/)
- On the misinterpretation of p-values: (http://www.onthelambda.com/2013/10/02/on-the-misinterpretation-of-p-values/)