Speeding Up R: Break out groups

Marco D. Visser Sean McMahon Caspar Hallmann

October 2, 2015

Workshop schedule

1. General introduction
2. Identifying whether and what to optimize
3. *Break out groups*

Workshop schedule

1. Parallel algorithms 101
2. Extending R with C
3. Own code

I: Parallel algorithms 101

Exercises

1. Review pitfalls (section 3)
2. How many cores do I have? (section 3.3)
3. Random numbers (section 3.4)
4. Build parallel bootstrap (section 3.4 or 3.5)

II: Extending R with C

Exercises

1. .C interface (section 5)
2. .Call interface (section 6.2)
3. SEXP (section 6.2.2)
4. PROTECT, UNPROTECT and gc() (sect. 6.2.2)
5. random number generation and other functions (sect. 6.2.3)

Wrap up: Tips

1. tips on finding source code
2. "inline"

III: Own code

1. Use smaller problem
2. Profile and find bottlenecks
3. Discuss
4. Repeat 2 & 3 if needed

remember to save all intermediate steps

II: Wrap Up

Tips on finding source code

```
page(lm)
```

```
}
 else {
     x <- model.matrix(mt, mf, contrasts)
     z <- if (is.null(w))
         lm.fit(x, y, offset = offset, singular.ok = singular.ok,
             ...)
     else lm.wfit(x, y, w, offset = offset, singular.ok = singular.ok,
         ...)
 }
```

II: Wrap Up

Tips on finding source code

```
sum

## function (..., na.rm = FALSE)  .Primitive("sum")
```

.Primitives and *.Internals* are found in /src/main/names.c

II: Wrap Up

Tips on finding source code Package pryr

```
pryr::show_c_source(.Primitive("sum"))

## sum is implemented by do_summary with op = 0
## Please visit
https://github.com/search?q=SEXP%20attribute_hidden%20do_summary+repo:wc
```

Only works for *.Primitives* and *.Internals*

II: Wrap Up

Print the function *cor*

```
else if (na.method != 3L) {
     x <- Rank(x)
     if (!is.null(y))
         y <- Rank(y)
     .Call(C_cor, x, y, na.method, method == "kendall")
 }
```

II: Wrap Up

```
getAnywhere('C_cov')


## A single object matching 'C_cov' was found
## It was found in the following places
##   namespace:stats
## with value
##
## $name
## [1] "cov"
##
## $address
## <pointer: 0x345e6c0>
## attr(,"class")
## [1] "RegisteredNativeSymbol"
##
## $dll
## DLL name: stats
## Filename: /usr/lib/R/library/stats/libs/stats.so
## Dynamic lookup: FALSE
##
## $numParameters
## [1] 4
##
## attr(,"class")
## [1] "CallRoutine"      "NativeSymbolInfo"
```

can be found in */src/library/stats/src/cov.c*

II: Calling C

```
#include <R.h>
#include <Rmath.h>
void foo(int *a, double *b) {
}
--- C code here ---
}
```

```
system("R CMD SHLIB -o foo.so foo.c")
```

```
dyn.load("Simple.so")
```

```
CWrapper <- function(a,b){
out <- .C("SimpleC",
a= as.integer(a),
b=as.double(b))
)
return(out)
}
```

II: inline package

```
require(inline)

## Loading required package:  inline
## Loading required package:  methods

foo<-cfunction(c(a="numeric"),
"
SEXP result = PROTECT(allocVector(REALSXP,1));
REAL(result)[0]=asReal(a) * asReal(a);
UNPROTECT(1);

return result;
"
)
foo(4)

## [1] 16
```