Running R Faster

Tomas Kalibera
My background: computer scientist, R user.
My FastR experience: Implementing a new R VM in Java.

- New algorithms, optimizations help
  - Frame representation, variable lookup
  - Function calls and argument matching
  - Specialized data types
  - Code specialization
  - Lazy arithmetics with profiling views

- Implementing a new R VM is hard
  - Specification
  - Tightly coupled packages and the VM

VEE’14: A fast abstract syntax tree interpreter for R
My current work: speeding-up GNU-R.

With Luke Tierney, Jan Vitek

ML benchmarks from TU Dortmund

github.com/kalibera/rexp

Based on R-dev 65969 (June 18), check-all passes.
Shootout benchmarks

Relative Speedup (gmean=1.92x)
Compiler bytecode-optimizations.

- Inlining constants into bytecode
- Inlining labels into bytecode

```javascript
function(x) {
  for(i in 1:10) { x <- x + 1 }
  x
}
```

```
1: 1:10,
2: i,
3: for (i in 1:10) { x <- x + 1 },
4: x,
5: 1,
6: x + 1
```

```
LDCONST.OP, 1L,
STARTFOR.OP, 3L, 2L, 16L,
7: GETVAR.OP, 4L,
  LDCONST.OP, 5L,
  ADD.OP, 6L,
  SETVAR.OP, 4L,
  POP.OP,
16: STEPFOR.OP, 7L,
  ENDFOR.OP,
  POP.OP,
  GETVAR.OP, 4L,
  RETURN.OP
```
Compiler optimizations – variable access.

- Special instruction for creating a promise that just reads a variable
  - Faster variable access for builtins (uses cache)
- Constant-pool re-ordering
  - Variable are first, which reduces memory overhead of the binding cache and improves locality

Frames in R are implemented using linked lists. A binding cache stores, for each constant in the constant pool, a reference to the corresponding element of the linked list.
Stack-allocation of call arguments. (primarily in the compiler)

- Call arguments passed as linked-list
- Special stack-based memory region
  - Growable, shrinkable stack for fixed-size call argument cells
  - Special treatment by the GC
- Support for long-jumps via contexts
- Better locality, faster reclamation

In R, the list of function arguments (promises) passed to a function are kept around for the duration of the function call, because they'll become needed in the case of object dispatch.
Explicit argument passing (no linked lists).

- For (many) builtins and internals
- For closures called positionally
  - Lists are only created lazily if needed

```r
get(x, envir, mode, inherits)
```

```r
SEXP attribute_hidden do_get(SEXP call, SEXP op, SEXP args, SEXP rho)
  if (!isValidStringF(CAR(args)))
  if (TYPEOF(CADR(args)) == REALSXP)
  if (isString(CADDR(args)))
    ginherits = asLogical(CADDDR(args));
```

```r
do_earg_get(SEXP call, SEXP op, SEXP arg_x, SEXP arg_envir, SEXP arg_mode, SEXP arg_inherits, SEXP rho)
```
Inlining wrappers to foreign calls.

```r
rnorm <- function (n, mean = 0, sd = 1)
  .External(C_rnorm, n, mean, sd)
```

- Inlining avoids overhead of promise creation, argument matching, environment creation
- Explicit passing of arguments to `.Call` foreign calls (avoiding linked list)
- Updating external pointer at load time

C_rnorm in the example is a variable in the `stats` namespace, which is automatically created when `stats` package is loaded and it points to a registered native symbol (R object). This object contains an external pointer (R structure), which contains a physical pointer to the `rnorm` routine implemented in the C code of the `stats` package.
Object dispatch (S3/S4) optimizations.

- Faster signature creation
  - Avoid name allocation
  - Re-use hashcode of first term “method”
  - Comparison using == (instead of strcmp)

- Fast-path optimizations

During method dispatch, one needs an R symbol for a signature (S3 or S4). A symbol has to be looked up in a hash table, based on its string name. Strings in R are however also interned (STRSXPs), and remember their hashes.
Summary

● GNU-R performance for real applications can be improved without changing current semantics
  – Avoiding linked lists for function arguments
  – Optimizing dispatch of stats functions, S3/S4 dispatch
  – Optimizing string operations
  – Smaller clean-ups (symbol, charsxp shortcuts, etc)
● I'm working with Luke Tierney on merging some of these improvements