Optymalizacja R – dlaczego warto przesiąść się na Linuxa?

Zygmunt Zawadzki

19 listopada 2014

Więcej informacji, wraz z dodatkowymi materiałami można znaleźć w repozytorium na GitHubie pod adresem `https://github.com/zzawadz/Prezentacja_ERKA_2014`. Ewentualne uwagi i pytania można kierować na adres `zawadzkizygmunt@gmail.com`.

Rzecz o szybkości R

R taki wolny...

	Fortran gcc 4.8.1	Julia 0.2	Python 2.7.3	R 3.0.2	Matlab R2012a	Octave 3.6.4	Mathe- matica 8.0	JavaScript V8 3.7.12.22	Go go1
fib	0.26	0.91	30.37	411.36	1992.00	3211.81	64.46	2.18	1.03
parse_int	5.03	1.60	13.95	59.40	1463.16	7109.85	29.54	2.43	4.79
quicksort	1.11	1.14	31.98	524.29	101.84	1132.04	35.74	3.51	1.25
mandel	0.86	0.85	14.19	106.97	64.58	316.95	6.07	3.49	2.36
pi_sum	0.80	1.00	16.33	15.42	1.29	237.41	1.32	0.84	1.41
rand_mat_stat	0.64	1.66	13.52	10.84	6.61	14.98	4.52	3.28	8.12
rand_mat_mul	0.96	1.01	3.41	3.98	1.10	3.41	1.16	14.60	8.51

Figure: benchmark times relative to C (smaller is better, C performance = 1.0).

C compiled by gcc 4.8.1, taking best timing from all optimization levels (-O0 through -O3). C, Fortran and Julia use OpenBLAS v0.2.8. The Python implementations of `rand_mat_stat` and `rand_mat_mul` use NumPy (v1.6.1) functions; the rest are pure Python implementations. Benchmarks can also be seen here as a plot created with Gadfly.

Rysunek : Źródło: `http://julialang.org/`

... inne języki takie szybkie (oprócz Matlaba).

rly?

Rzecz o szybkości R - cd.

Rzeczywiście R nie jest demonem szybkości i można pisać w nim BARDZO nieefektywny kod:

```
library(microbenchmark)
x = 1:10000
slow_sum = function(x)
{
  sum = 0
  for(i in 1:length(x)) sum = sum + x[i]
  sum
}
microbenchmark(sum(x), slow_sum(x))

## Unit: microseconds
##         expr      min        lq       mean   median        uq      m
##       sum(x)   12.320   12.5055   15.49794   16.530   18.1945   22.4
##  slow_sum(x) 4159.101 4271.8230 4642.03785 4395.957 5066.3590 6043.6
##  neval
##    100
##    100
```

Dobrym pytaniem byłoby - "ale dlaczego R jest taki wolny?"
R w zamierzeniach nie musiał być demonem szybkości. R, a w zasadzie jego poprzednik S, miał być głównie interfacem do kodu fortranowego (ewentualnie innego języka). A sam interface nie musi być bardzo szybki, bylby tylko był wygodny.
Więcej - `https://www.youtube.com/watch?v=_hcpuRB5nGs` - prezentacja twórcy języka S - Johna Chambersa (również członka R Core Team), na tamat przeszłości i przyszłości R.

Zagadka!

W R występują trzy główne operatory wyłuskania wartości:

```
x[1]
x[[1]]
x$a
```

Sam R został napisany w C. Pytanie brzmi: ile linijek kodu ma plik w którym te operatory są zdefiniowane?

Zagadka! Rozwiązanie.

Plik ten można znaleźć w źródłach R w `/src/main/subset.c`.

Posiada on 1288 linijek...

Tyle kodu ma jednak sens - w nim zawarte jest między innymi obsługiwanie częściwego dopasowywania nazw w liście:

```
x = list(dluga_nazwa = 1:5, inna_nazwa = 1:5)
x$dlu # dziala!

## [1] 1 2 3 4 5
```

Ma to jednak oczywiste konsekwencje wydajnościowe.

Realizacja idei interface w praktyce - obliczenia macierzowe

W praktyce jednak wydajność R nie jest znaczącym problemem - w końcu miał być głównie interfacem dla innych języków i bardzo dużo kodu w R służy jedynie wywołaniu określonej biblioteki, która wykonuje najcięższe obliczenia.
Np. funckja eigen - służąca obliczaniu wartości własnych i wektorów własnych korzysta z biblioteki do obliczeń macierzowych LAPACK (Linear Algebra PACKage), która została napisana w Fortranie. Można to poznać po tym, że wewnątrz funkcji znajduje się linijka wywołująca kod zewnętrzny:

```
.Internal(La_rs(x, only.values)) # z eigen
.Call(...) # inny interface - do C i C++
.External(...)
.C(...) # stary interface do C
```

Definicję La_rs można znaleźć w `/src/modules/lapack/Lapack.c`, a szukając dalej w tym folderze można również znaleźć kod fortranowy Lapack'a (ok 200k linii kodu - duża rzecz...).

Lapack i BLAS

Gdyby wejść głębiej, okaże się, że LAPACK (kod fortranowy), korzysta z implementacji BLASu zawierającej zbiór podstawowych operacji na macierzach takich jak np. mnożenie macierzowe. Sam BLAS to API (Application Programming Interface), opisujące jak powinna wyglądać biblioteka do obliczeń macierzowych. Na podstawie tego opisu tworzy się implementacje które realizujące te obliczenia.

Istnieje kilka implementacji BLASu - np. ATLAS, MKL i OpenBLAS. R również posiada swoją implementację, jednak nie jest ona najszybsza...

... skoro ten standardowy R-owy BLAS nie jest najszybszy, to może go podmienić na wydajniejszy?

... skoro ten standardowy R-owy BLAS nie jest najszybszy, to może go podmienić na wydajniejszy?
Zaraz będzie opis jak to prosto zrobić na Linuxie...

... skoro ten standardowy R-owy BLAS nie jest najszybszy, to może go podmienić na wydajniejszy?
Zaraz będzie opis jak to prosto zrobić na Linuxie...

... ale coś za dużo tekstu w tej prezentacji. Dalszy tekst będzie nudny...

... skoro ten standardowy R-owy BLAS nie jest najszybszy, to może go podmienić na wydajniejszy?
Zaraz będzie opis jak to prosto zrobić na Linuxie...

... ale coś za dużo tekstu w tej prezentacji. Dalszy tekst będzie nudny...

... może by tak pokazać na żywo, jak to się robi?

Całe działanie wzorowane jest na wpisie z
`http://www.stat.cmu.edu/~nmv/2013/07/09/for-faster-r-use-openblas-instead-better-than-atlas-trivial`

... skoro ten standardowy R-owy BLAS nie jest najszybszy, to może go podmienić na wydajniejszy?
Zaraz będzie opis jak to prosto zrobić na Linuxie...

... ale coś za dużo tekstu w tej prezentacji. Dalszy tekst będzie nudny...

... może by tak pokazać na żywo, jak to się robi?

Całe działanie wzorowane jest na wpisie z
`http://www.stat.cmu.edu/~nmv/2013/07/09/for-faster-r-use-openblas-instead-better-than-atlas-trivial`

Czas na konsolę!

... mam nadzieję, że prezentacja optymalizacji BLAS na żywo się udała.

... mam nadzieję, że prezentacja optymalizacji BLAS na żywo się udała.
Jeżeli tak, to znaczy, że konsola nie jest taka straszna i można w niej troszkę poczarować:)

Co dalej?

Nie dawno Revolution Analytics wprowadziło swoją wersję R (`http://mran.revolutionanalytics.com/download/`), która wykorzystuje MKL - czyli BLAS stworzony przez Intela - jest on naprawdę szybki i być może sztuczka z podmianą R-owego BLAS-a przestaje mieć jakiekolwiek znaczenie w przypadku używania wersji od Revolution.

Trzeba to sprawdzić!

Dlaczego warto przejść na Linuxa?

- System bardzo konsolowy - a co za tym idzie - bliższy R.
- Nie ma Office:)
- Czasem dosyć problematyczny, a co za tym idzie frustrujący (a to dobrze! `http://youtu.be/JxwxefRAu70?t=30m` - wywiad z jednym z mistrzów R, nt. wartości frustracji).

Dziękuję za uwagę!