parallelquicksort

CarlosVargas

November 24, 2021

Quicksort Activity Report

```
# The environment
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
```

The purpose of this report is to show how sensible the results of an experiment could be to small changes in the testing parameters.

We use this notebook as the base of our analysis.

Machine Specifications

Our machine is a Dell Inspiron 15 3000 with an `AMD Ryzen 5 3500U` and 8 cores.

First attempt with original code

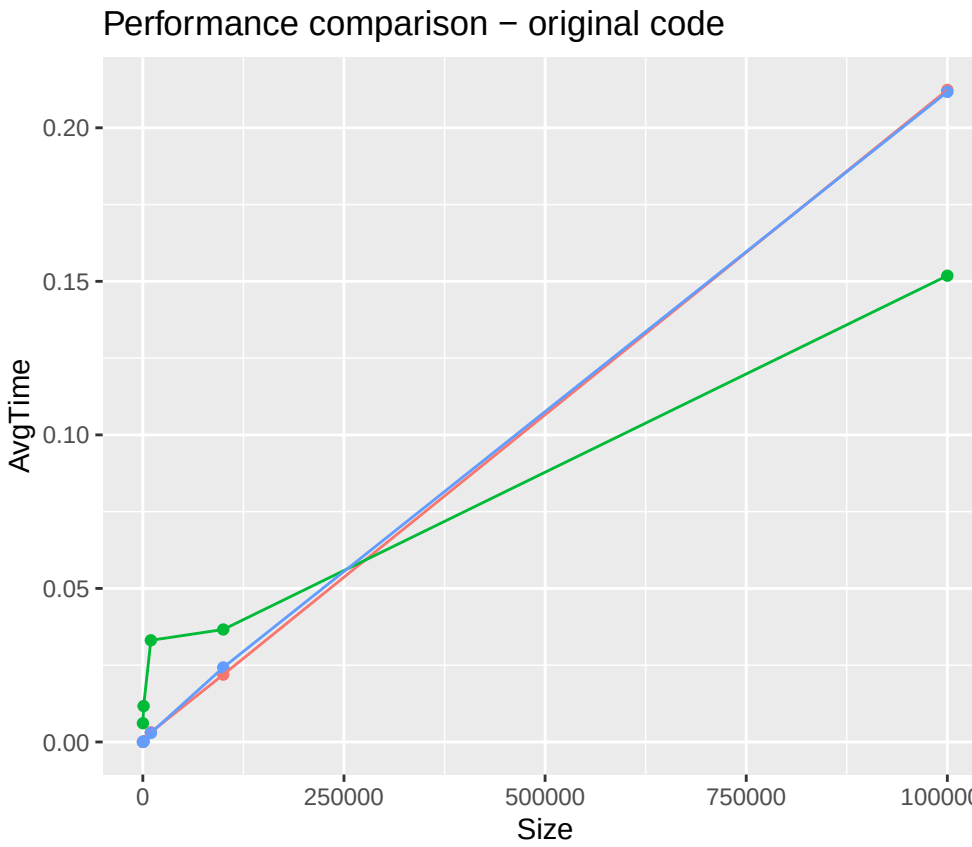
We launch the experiments with usign the base code in the notebook. After compilation, we run `run_benchmarking.sh` file to obtain the first round of results.

We also used the provided perl command to parse the data into a csv file. Then we create a dataframe grouping the Type and Size attributes adn computing the average over the runs.

```
df <- read.csv("data/fran-pc_2021-11-24/measurements_18:48.csv",header=T)

new_df = df %>% group_by(Type, Size) %>% summarise(AvgTime=mean(Time))
```

```
## `summarise()` has grouped output by 'Type'. You can override using the `.groups` argument.
```

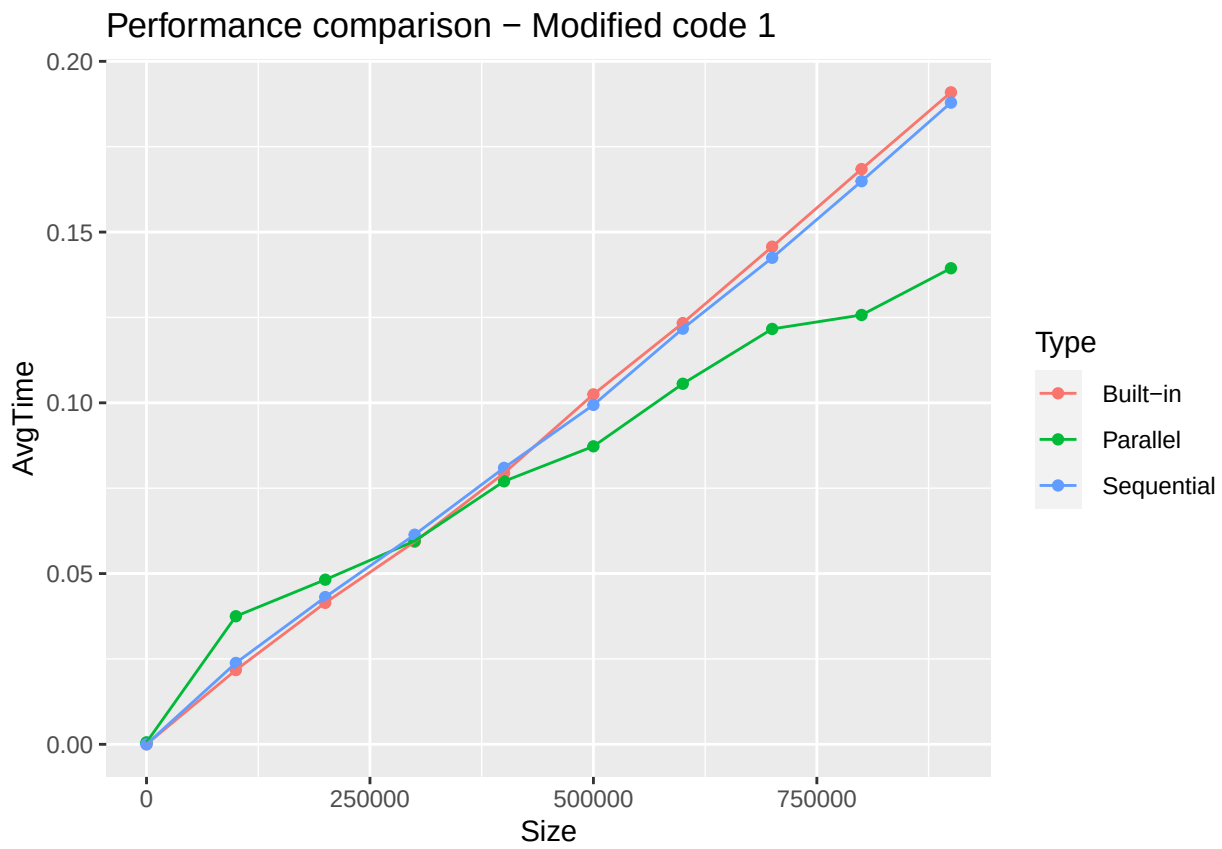


We plot the initial results using `ggplot`.

In this first result we can observe that for this specific machine, the parallel version is faster than the other two at an array size of about 270000. The initial analysis conducted by Arnaud Legrand concluded that the critical size was about 400000.

The original code propose unbalanced array sizes, so there is a big gap among the observations. We decided to add more points in the middle of the range and perform again the analysis. For this, we modify the `run_benchmark.sh` so that it generates

```
## `summarise()` has grouped output by 'Type'. You can override using the `.groups` argument.
```



Now, we have a better picture of the behaviour of the performances over the range of sizes.

Let's use now the option `-O3` in the makefile, which should be more convenient for performance testing.

```
...
CFLAGS = $(WARNINGS) -O3 -pthread -lrt -std=c99
...
```

Then repeat the test.

```
## `summarise()` has grouped output by 'Type'. You can override using the `.groups` argument.
```

Now,the plot shows that the parallel implementation is lower than the sequential until the array size reachs about 875000. This is a big change compared with previous tests.

Conslusion

As we have demonstrated, the initial configuration (hardware) as well as the different parameters (e.g. optimization methods at compiler time) have a drastical impact on the observed results. It is really important to pay attention to those factors in order to garantee reproducibility. The design specification of the experiments is also crucial to show up hidden patterns and behaviours in the data, so several different scenarios shuld be considered.