Laboratory Notebook for a Multi-Threaded Version of Quicksort (Updated)

Vu Anh Tuan

2025-10-08

Project Overview

This project aims at providing an efficient multi-threaded implementation of the QuickSort algorithm on multi-core machines. This document contains my attempts to evaluate the performance of an implementation of such code.

General Organization

src/

This directory comprises the parallel implementation and a standard Makefile to compile it.

data/ (Updated - VU Anh Tuan)

This is where raw experimental data should go. Each directory entry comprises a set of experiments and the directory name is based on the machine name and on the date.

```
echo mkdir data/`hostname`_`date +%F`
```

```
## mkdir data/THINKBOOK_2025-10-08
```

Create a directory for experiment data: `mkdir data/<hostname>_<date>`

Notes: This command may be omitted, as the subsequent commands will automatically create the necessary directory during execution. However, to ensure a smooth workflow, it is recommended to create the file beforehand.

Typical usage

Compilation

A simple makefile with various compilation options is provided in the `src/` directory. Compilation is thus done by running `make -C src/`.

Compilation produces the parallel Quicksort executable along with object files.

Running the code

The code is simple and can be run with `./src/parallelQuicksort [array_size]`.
By default, `array_size` is 1,000,000.

The code executes and sorts the array using:

1. A custom sequential implementation.
2. A custom parallel implementation.
3. The built-in `libc` qsort function.

Times are reported in seconds.

Experimental Reports

2025-10-08 (VU Anh Tuan)

Initial code executation

- The initial implementation was obtained from SC12 HPC Educator. The original author is Joshua Stough from Washington and Lee University and the modified version from Arnaud Legrand Github.
- Typical first execution on my laptop (Intel i5-12500H, Ubuntu 24.04.3 LTS Linux 6.14.0-33-generic) showed ***the sequential version ran faster than the parallel one.***

```
./src/parallelQuicksort
```

```
## Sequential quicksort took: 0.119889 sec.
## Parallel quicksort took: 0.159347 sec.
## Built-in quicksort took: 0.123528 sec.
```

Then, I follow the instruction of first series of experiments. I ran the three algorithms with varying array sizes, repeating each measurement 5 times.

```
bash ./scripts/run_benchmarking.sh
```

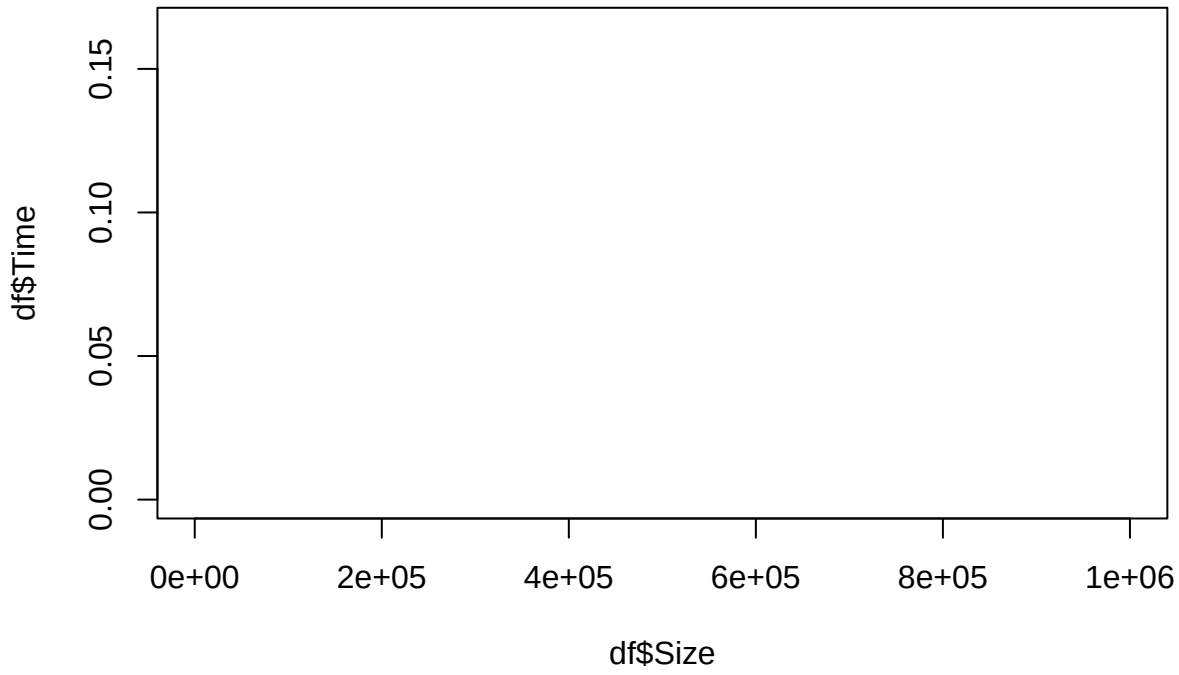
Experimental results were saved in `.txt` files for further analysis.

A substep should be done before plotting to parse `.txt` file to `.csv` file:

```
perl ./scripts/csv_quicksort_extractor.pl < data/THINKBOOK_2025-10-08/measurements_17\:43.txt > data/TH
```

Parsed results were visualized using R.

```
df <- read.csv("data/THINKBOOK_2025-10-08/measurements_17:43.csv",header=T)
plot(df$Size,df$Time,col=c("red","blue","green")[df$Type])
```



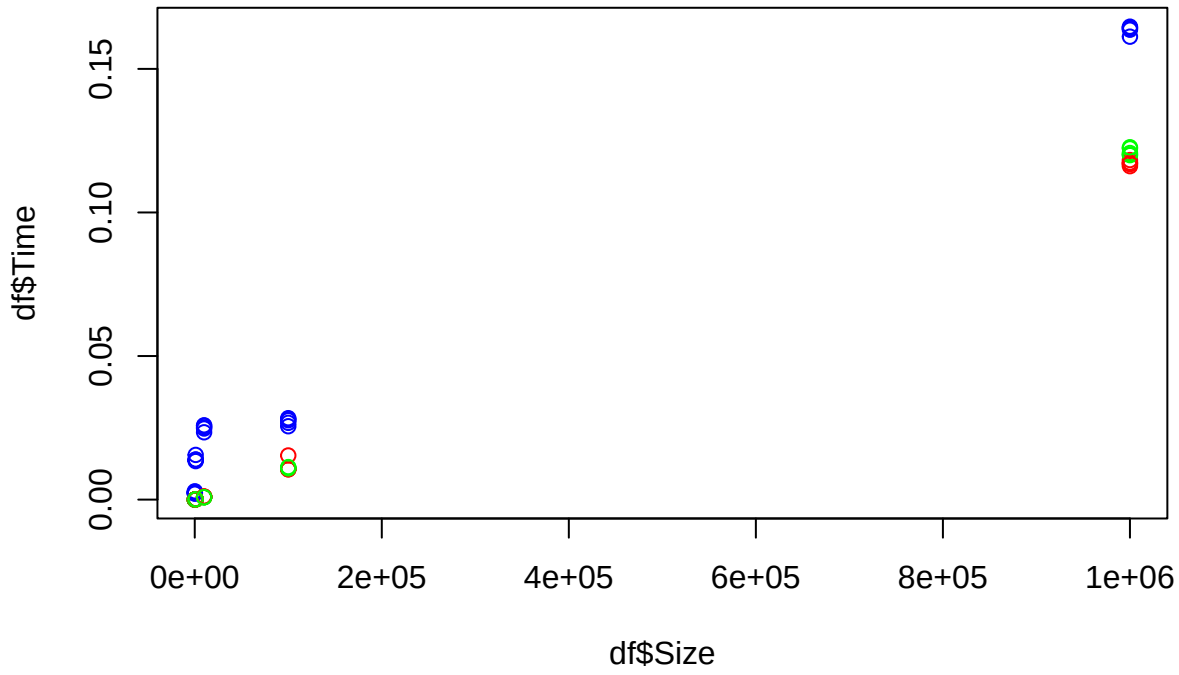
Unfortunately, there is no point being plotted in the figure.

I noticed that the variables in the `Type` column are of type `chr` rather than `int`. As a result, the parameter `col = c("red","blue","green")[df$Type]` cannot automatically assign colors. I made a minor adjustment by using a dictionary `colors` to map specific colors to each value.

```
colors <- c(" Sequential" = "red", " Parallel" = "blue", " Built-in" = "green")
```

Then, run again.

```
df <- read.csv("data/THINKBOOK_2025-10-08/measurements_17:43.csv",header=T)
colors <- c(" Sequential" = "red", " Parallel" = "blue", " Built-in" = "green")
plot(df$Size,df$Time,col=colors[df$Type])
```



The points now appear in the figure

A basic plot shows how execution time varies with array size for the sequential, parallel, and built-in versions.

The plot shows that the sequential version is faster than the parallel one for larger arrays.

An alternative visualization using gnuplot.

```
FILENAME="data/THINKBOOK_2025-10-08/measurements_17:43"
perl scripts/csv_quicksort_extractor2.pl < "$FILENAME.txt" > "${FILENAME}_wide.csv"
echo "
  set terminal png size 600,400
  set output '${FILENAME}_wide.png'
  set datafile separator ','
  set key autotitle columnhead
  plot '${FILENAME}_wide.csv' using 1:2 with linespoints, '' using 1:3 with linespoints, '' using 1:4 w
" | gnuplot
echo [[file:${FILENAME}_wide.png]]
```

```
## [[file:data/THINKBOOK_2025-10-08/measurements_17:43_wide.png]]
```

Notes: Check whether the gnuplot package exists before execution, as it is not automatically installed on Ubuntu 24.04.3 LTS. A base package is enough to use in this project.

```
sudo apt update
sudo apt install gnuplot-nox
```

Conclusion: I noticed that my results differ somewhat from those provided in the instructions. This variation is probably due to differences in CPU architecture, as I am using an i5 processor (while the instructions were based on an i7).