Merge Sort

Erik Saule

Goal: Implement a merge sort algorithm in C++ and benchmark it on Centaurus.

1 Programming

For reference, here is a primer on merge sort according to geeks for geeks.

Merge sort is a popular sorting algorithm known for its efficiency and stability. It follows the divide-and-conquer approach to sort a given array of elements.

Here's a step-by-step explanation of how merge sort works:

- Divide: Divide the list or array recursively into two halves until it can no more be divided.
- Conquer: Each subarray is sorted individually using the merge sort algorithm.
- Merge: The sorted subarrays are merged back together in sorted order. The process continues until all elements from both subarrays have been merged.

Or alternatively, here is a youtube video. `https://www.youtube.com/watch?v=4VqmGXwpLqc`

There is a set of crash course on how to program in C++ in Canvas.

TODO. Implement merge sort of an array of integers in C++. Implement it yourself from scratch. It is good training to do that yourself from algorithm description. Provide necessary tooling to run on different data size. Printing the time it took to sort the array on `stdout` or `stderr`. Make it easy to run and check. Write in a README file instructions on how to run it.

Suggestions. I suggest taking the size of the array as a command line parameter for easy testing. Generate the data randomly. I suggest writing the code in `git` so you can easily share with instruction team for feedback in case of bug. Provide a `Makefile` that is correct and generic so whoever tests it can just `make clean; make` and go ahead and test.

2 Benchmark on Centaurus

We'll benchmark the code on Centaurus. Check Canvas for details on how to access Centaurus. Note that when you ssh on Centaurus you are on the head node, not a compute node. You should not run expensive code on the head node. (Compile, write scripts, plot results. but don't run memory hungry code or compute hungry code on the head node.)

You'll need to write a script that runs the code you need to run. And run that script through SLURM which will run the script in a particular node.

TODO. Benchmark the code on a computing node of Centaurus. Benchmark sizes for all power of 10 from 10 to 10^9. Report time. Also plot a chart. Document how to benchmark in README and comment on whether the times you get make sense in README.

Suggestions. Remember that binaries are possibly not compatible from one Linux system to another. So you will need to recompile the code on Centaurus. (So probably gitignore the binaries and object files.) I'd write the code to write the size and time to a file in an easy to parse format. And then plot the data in that file. Plot chart in logscale. Remember to label axes. Plot chart using python matplot lib; that's likely the easiest reproducible thing to do.

3 Submit

TODO. Make an archive of all code file and result file. (You can skip the object files and binaries; but include code files, scripts, output plots.)