departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Concurrency and Parallelism 2018-19

(4) Parallel Sort

João Lourenço

October 3, 2018

Abstract

In this class you will learn about Cilk+ and improve your knowledge on parallel patterns and on the development of parallel programs.

1 Introduction

The Quick Sort algorithm uses divide and conquer to gain the same advantages as the merge sort, while not using additional storage. As a trade-off, however, it is possible that the list may not be divided in half. When this happens, we will see that performance is diminished.

Google for a quick tutorial on Quick Sort if you need to remember how it works.

2 Lab Work

2.1 Given Version

You may find at

`https://joaomlourenco@bitbucket.org/cp201819/quick-sort.git`

a running sequential version of the Quick Sort algorithm.

Clone the given version in your Linux device (own laptop, lab workstation, or as a last resort, in the "node9" server used in the last lab class) with

```
git clone https://joaomlourenco@bitbucket.org/cp201819/quick-sort.git
```

and compile it using the command `make`. Make is a command that builds a project given in a project specification file named `Makefile`. Have a look at this text file and learn a bit about `make` and `Makefile`. You'll need it again in the short future.

The command `make` will build both the sequential and the parallel versions of the Quick Sort algorithm, but currently the parallel function is empty (does nothing).

Try running the sequential version for different sizes of the array to be sorted.

2.2 Work plan

The given version of the Quick Sort algorithm is sequential.

Your job is to make a parallel version of this algorithm (in file `qs-parallel.c`) using Cilk+!

Please follow these steps:

1. Clone the given project.
2. Compile and experiment with the given version. Study the source code and understand how it works.
3. Implement the `qsort_parallel()` function in the `qs_parallel.c` file.
4. Experiment with different array sizes. What can you conclude about the relative speed of the *seuqential* vs. the *Cilk+* versions?

Please remember to use GIT appropriately. Leran to work with branches and make a branch when you start a new phase from the list above, and merge your branch to your main version when the development is finished and appropriately tested/validated.

Acknowledgments

The text from the Introduction is an adaptation from the text in `http://interactivepython.org/courselib/static/pythonds/SortSearch/TheQuickSort.html`.