Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Parallel Programming
Assignment 5: Divide and Conquer
Spring Semester 2020

Assigned on: **18.03.2020**

Due by: **(Wednesday Exercise) 23.03.2020**
(Friday Exercise) 25.03.2020

Overview

In this assignment we will implement a multithreaded version of a sequential algorithm by using the Divide and Conquer design pattern. Furthermore, we review basic theoretical concepts of parallel programming including Amdahl's Law and Gustafson's Law.

Getting Prepared

- Download the ZIP file named `assignment5.zip` on the course website.
- Import the project in Eclipse: Click on *File* in the top-menu, then select *Import*. In the dialog, select *Existing Projects into Workspace* under the *General* directory, then click on Next. In the new dialog, select the radiobox in front of *Select archive file* to import a ZIP file. Then, click Browse on the right side of the text-box to select the ZIP file you just downloaded from the website (`assignment5.zip`). After that, you should see assignment5 as a project under *Projects*. Click Finish.
- If you have done everything correctly, you should now have a project named assignment5 in your *Package Explorer*.

1 Search and Count

When hiring new people for a job position, the first task is the screening of CVs. The Human Resources (HR) manager has to read all the CVs which have been previously validated by the automatic CV checker. The manager checks whether the applicants have a specific set of required qualities. If the applicant is qualified, then they mark those CVs as being valid for a certain job. However, some features are easier to check than others. For example, it is easier to get the age of the candidate based on the birth year, than to evaluate if the candidate has the necessary technical and social skills a job requires. We can say that the first task requires a *light* workload for each candidate, while the second task requires a *heavy* workload.

In our case, the CVs for the HR manager are randomly generated numbers and the desired features can be faster or slower to compute. We classify two types of workloads: `HEAVY` and `LIGHT` (e.g. check if the number is non zero - light workload, and check if the number is prime - heavy workload). The code to check the features is given to you and should not be changed (it is located in the class `Workload` in the `doWork` function). The task is to count for each feature how often it appears in the given input.

The HR manager knew about the advantages of automating the CV screening process and bought a single threaded program to solve this task. The relevant code is located in the `SearchAndCountSeq` class. But with a rising number of applicants the HR manager hopes that the program can be made faster.

Task A: The HR manager hired you to speed-up the screening process by creating a parallel version of the program. As a first step, you need to decide how to split the problem into smaller tasks to be solved in parallel. For this purpose rewrite the sequential version using the Divide and Conquer design pattern:

```
Divide and Conquer:
  if cannot divide:
    return unitary solution (stop recursion)
  divide problem into two
  solve first (recursively)
  solve second (recursively)
  return combine solutions
```

Complete the sequential implementation using the Divide and Conquer design pattern in the provided skeleton class `SearchAndCountSeqDivideAndConquer`. Make sure that your implementation is correct.

Task B: Solving our problem using the Divide and Conquer design pattern gives us a straightforward way to make it parallel – instead of executing the two tasks sequentially we execute them in parallel. Implement a parallel version of `SearchAndCountSeqDivideAndConquer` in the provided skeleton class `SearchAndCountThreadDivideAndConquer`.

Extend your implementation such that it creates only a fixed number of threads regardless of the problem size. Make sure that your solution is properly synchronized when checking whether to create a new thread.

Task C: Next, the HR manager wants you to show them that your program indeed works faster. Compare the runtime of you parallel implementation with the original sequential version. To improve the runtime of your parallel implementation implement a sequential `cutOff`, that is, use the sequential version to process inputs smaller than the `cutOff` value instead of recursing always to the base case that is not divisible (i.e., processing a single element which is equivalent to the `cutOff` value 1).

Further, your manager does not want to meddle with its settings. Find the value for the `cutOff`, and the number of threads that maximize the speedup for input size $100,000$ and for both workloads.

Task D (Optional): Instead of creating threads manually, adapt your code to use `ExecutorService`.

2 Amdahl's and Gustafson's Law

Assuming a program consists of 50% non-parallelizable code.

a) Compute the speed-up when using 2 and 4 processors according to Amdahl's law.

b) Now assume that the parallel work per processor is fixed. Compute the speed-up when using 2 and 4 processors according to Gustafson's law.

c) Explain why both speed-up results are different.

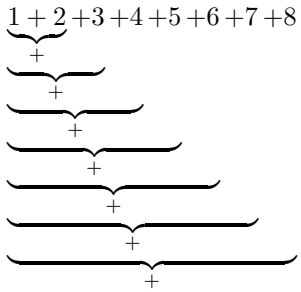
3 Amdahl's and Gustafson's Law II

a) The analysis of a program has shown a speedup of 3 when running on 4 cores. What is the serial fraction according to Gustafsons law?

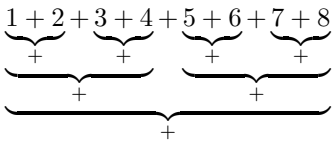
b) The analysis of a program has shown a speedup of 3 when running on 4 cores. What is the serial fraction according to Amdahls law (assuming best possible speedup)?

4 Task Graph

Assuming you want to add eight numbers, then two options to do this are



and



a) Given those two variants, determine the length of the critical path for both computations.

b) For a sequence of length n, determine the length of the critical path using the two approaches from above (accumulator method and Divide and Conquer).

Submission

In order to receive feedback for your exercises, you need to submit your code to the Git repository. You will find detailed instructions on how to install and set-up Eclipse for use with Git in Exercise 1.

Once you have completed the skeleton, commit it to Git by following the steps described below. For the questions that require written answers, please write them on paper and bring them to the next exercise session where the solutions will be discussed.

- **Check-in your project for the first time**
  - Right click your created project called **assignment5**.
  - In the menu go to **Team**, then click **Share Project**.
  - You should see a dialog Configure Git Repository. Here, next to the Repository input field click on **Create...**
  - Select a root git directory or your projects that you have created in Execise 1. Note for all your assignments you should use the same directory.
  - click **Finish**.
- **Commit changes in your project**
  - Now that your project is connected to your git repository, you need to make sure that every time you change your code or your report, at the end you commit your changes and send (push) them to the git server.
  - Right click your project called **assignment5**.
  - In the menu go to **Team**, then click **Commit...**.
  - In the Comment field, enter a comment that summarizes your changes.
  - In the Files list, select all the files that you changed and want them to be committed. This typically includes all the Java files but not necessarily all the files (e.g., you dont have to commit setting files of our eclipse installation).
  - Then, click on **Commit** to store the changes locally or **Commit and Push** to also upload them to the server. Note that in order to submit your solution you need to **both** commit and push your changes to the server.
- **Push changes to the git server**
  - Right click your project called **assignment5**.
  - In the menu go to **Team**, then click **Push Branch 'master'**. Note if this is not your fist push you can also use **Push to Upstream** to speed up the process.
  - A new dialog appears, now fill in for the URL field:
    `https://gitlab.inf.ethz.ch/COURSE-PPROG20/<nethz-username>.git`
  - Click **Next**
  - Keep the default values and click **Next**
  - An authentication dialog should appear. Fill in your nethz username and password and click **OK**.
  - Click **Finish** to confirm your changes. Note that eclipse might ask for authentication again.
- **Browse your repository online**
  - you can access and browse the files in your repository online on GitLab at:
    `https://gitlab.inf.ethz.ch/COURSE-PPROG20/<nethz-username>`