DEI-FCTUC

Computação de Alto Desempenho – 2012/2013
Exercises #2: Introduction to the package java.util.concurrent

The goal of this assignment is to know the package `java.util.concurrent`. For this you should use multiple threads to minimize the time it takes to do a multiplication of sparse matrices. To represent these matrices we will use the Compressed Row Storage (CRS) Format, briefly described here: http://netlib.org/utk/papers/templates/node91.html. Some constraints that you should pay attention to:

- Check the code to multiply sparse matrices that comes with this assignment. In the source code you have a PNG file with partial information about CRS.
- To synchronize the threads **you must not use Java monitors.**
- You should measure all the times in the exercises, to determine which options are best.

1. Consider a multi-threaded master-workers program, where each thread takes care of part of the multiplication (e.g., a subset of the lines of the product matrix). Use the `Executors` class to launch the threads. The main thread should get back the results using a `Future`. At start time, each thread should know its work (e.g., the lines it responsible for).

2. This time, the master should use an `ArrayBlockingQueue` to send the work to the workers. Each worker may compute more than a single part of the product matrix. One of the main problems with this approach is to gather the results from the threads. You may use locks from `java.util.concurrent` to let the threads modify a *shared* product matrix concurrently.

3. Now use a `ConcurrentHashMap` to store the results from the worker threads. Once the last thread finishes its task, the gathering thread should read the partial results from this hash map and take care of the product matrix alone. The new problem in this exercise is to let the master thread know that all the others finished. You may use a `CountDownLatch` for this purpose.

4. Now, you may assign each thread with its own piece of work without using any kind of master. You should compare the execution time of this solution, compared to the previous ones.