Lab 5: Data-Driven Futures and Phasers

Instructor: Mackale Joyner, Co-Instructor: Zoran Budimlić

Course Wiki: `http://comp322.rice.edu`

Staff Email: comp322-staff@mailman.rice.edu

Goals for this lab

- Experimentation with Data-Driven Futures with Abstract Metrics

Lab Projects

Today, we will use a Cholesky factorization example to learn about using Data Driven Futures. We will compute performance for this part of the lab using abstract metrics. Please run your tests locally, as there is no autograder assignment for this lab.

The Maven project for this lab is located in the following svn repository:

`https://svn.rice.edu/r/comp322/turnin/S18/`*`NETID`*`/lab_5`

For instructions on checking out this repository through IntelliJ or through the command-line, please see the Lab 1 handout. The below instructions will assume that you have already checked out the lab_5 folder, and that you have imported it as a Maven Project if you are using IntelliJ.

1 Parallelization of Cholesky (using Data-Driven Tasks)

In linear algebra, the Cholesky factorization is a decomposition of a positive-definite matrix into the product of a lower triangular matrix and its conjugate transpose. This decomposition is useful to efficiently obtain numerical solutions, e.g., for a system of linear equations of the form $A \times x = b$.

We have provided a sequential version of Cholesky Factorization in the `CholeskyFactorization.java` and `CholeskyFactorizationSequential.java` files. Your goal will be to parallelize this computation using data-driven tasks and data-driven futures, and to evaluate your parallelization using abstract metrics. The fact that the sequential version uses a generic `dataStore` container can be leveraged to simplify this conversion.

Your assignment is to create a parallel version of `CholeskyFactorizationSequential.runComputation()` that implements the Cholesky Factorization algorithm by using data-driven tasks:

1. Write a parallel version of the `runComputation()` method of the `CholeskyFactorizationParallel.java` file using data-driven tasks with calls to `asyncAwait()`. The provided file has helpful hints to guide you in this process.
2. For abstract metrics, note that `CholeskyFactorizationParallel.java` is already set up with zero overhead for creating async tasks, as in Homework 1 (but not Homework 2).
3. Run the unit tests locally to verify whether your parallel computation achieves the required abstract metrics.

2 Turning in your lab work

For lab_5, you will need to turn in your work before March 5, 2018 at 11:59 pm, as follows.

1. Show your tests passing locally to an instructor or TA to get credit for this lab.
2. Commit your work to your `lab_5` turnin folder. The only changes that must be committed are your modifications to `CholeskyFactorizationParallel.java`. Check that all the work for today's lab is in your `lab_5` directory by opening

   `https://svn.rice.edu/r/comp322/turnin/S18/NETID/lab_5/`

   in your web browser and checking that your changes have appeared.