Assignment 2 CS4745/6025 Fall 2017

Due: 9:00 am Friday October 13 electronically in Desire2Learn

1. A SIMD implementation of sum-reduction (Algorithm 4.2), using Cilk Plus, is given in `reductionCilkSIMD.c`, available at the textbook's GitHub site: https://github.com/eaubanel/ElementsOfParallelComputing.git. Modify this program to handle an array of any size, not just powers of two. This requires a few changes to the algorithm. First, the `for` loop starts at $k \leftarrow \lfloor \log n \rfloor$. Second, in Algorithm 4.2 the size of the group of operands involved in the additions shrinks by a factor of two each iteration ($i \in [0..2^k)$). In the modified algorithm the number of operands may be odd for one or more iterations. If this is the case the last element of the group is added to the second last element, then the SIMD loop can operate on an even number of operands.

   Change the input to the program to obtain the size of the array on the command line:

   ```
   if(argc <2){
       fprintf(stderr,"usage: %s n\n", argv[0]);
       return 1;
   }
   int n = strtol(argv[1], NULL, 10);
   ```

   Then compute $exp \leftarrow \lfloor \log n \rfloor$. This can easily be done with a loop of right shifts (`>>`).

   Submit a terminal session showing output for 3 different array sizes.

2. Matrix multiplication ($C \leftarrow A \times B$, where all are $n \times n$ matrices) can be computed using a divide-and-conquer strategy. The computation is split into four parts, by splitting each matrix in four:

$$\begin{bmatrix} C_{00} & C_{01} \\ C_{10} & C_{11} \end{bmatrix} = \begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix} \cdot \begin{bmatrix} C_{00} & C_{01} \\ C_{10} & C_{11} \end{bmatrix}$$

$$= \begin{bmatrix} A_{00} \cdot B_{00} + A_{01} \cdot B_{10} & A_{00} \cdot B_{01} + A_{01} \cdot B_{11} \\ A_{10} \cdot B_{00} + A_{11} \cdot B_{10} & A_{10} \cdot B_{01} + A_{11} \cdot B_{11} \end{bmatrix}$$

   This process continues recursively until the matrices are of size 1×1. Decompose this approach into independent tasks, and make a fork-join implementation in Cilk Plus, starting from the sequential code provided in D2L. Pay special attention to the data dependencies, and *do not create any additional arrays*. Your program should also perform iterative sequential matrix multiplication, both to verify the parallel version and to compare the runtimes.

   Submit a record of your terminal session, for 3 different values of n.

What to submit: a pdf report with code listings and terminal sessions and a zip file with both source files.