Samuel Palmer
Joe Elizondo
CS380P (Spring 2010)

Homework 1

Our Work

In addition to the prefix sum problem we implemented solutions for the team standings problem and the index of last occurrence problem as described on pages 117 and 118 of the textbook.

Our implementation utilizes a barrier so each thread will synchronize at each level of the tree on the way up and on the way down. This barrier prevents fast threads from attempting to read values that have not been computed by slower threads.

In addition to our parallel implementation using Pthread we implemented a sequential algorithm for testing purposes. teamStandingsGen and lastoccurGen generate test cases and provide expected output to test our parallel prefix algorithms against. The parallel prefix algorithm matches the expected output in all of our test cases.

Testing Methodology

Our algorithms teamStandingsGen and lastoccurGen are capable of quickly generating a large number of complex test cases and work as follows:

To generate a test case for teamStandings:

./teamStandingsGen [# of teams] [# of games] [array out file] [sequential algo. output file]

This generates a file specified by [array out file] where the first number is the number of teams, the second is the number of games, and the remaining numbers represent the winner of each game.

To generate a test case for lastoccur

./lastoccurGen [# of possible values k] [# of elements] [array out file] [sequential algo. Output file]

This generates a file specified by [array out file] where the first number is the number of possible values *k,* the second number is the number of elements in the array, and the remaining numbers are numbers from (1..*k*).

These algorithms made it easy to submit many large test cases to Ranger. To test for correctness on Ranger we generated five test cases for each of our algorithms that increased in range and size of the inputs. We decided to run each set of five test cases multiple times to test for race conditions and decided, due to time constraints, that ten submissions each for a total of 100 submissions, would have to suffice. Although most of our submissions did not complete due to wait time on Ranger we are still confident in the correctness of our algorithms based on the results we did observe.

Evidence of Correctness

To run teamStandings:

./teamStandings [data input file] [# of threads]

The data input file is of the form described below the teamStandingsGen explanation and the number of threads must be a power of 2 (2, 4, 8, 16, etc) because of the following line in the scan algorithm:

...

if(index % (2 * stride) == 0)
...

To run lastoccur:

./lastoccur [data input file] [# of threads]

The data input file is of the form described below the teamStandingsGen explanation and the number of threads must be a power of 2.

Simple Test Cases

Last Occurrence
Test Case 1 Input:
4
6
1 1 1 4 4 4
Output:
0 0 1 0 3 4

Test Case 2 Input:
5
7
1 4 4 3 3 1 1
Output:
0 0 1 0 3 0 5

Team Standings
Test Case 1 Input:
5
10
1 0 0 2 1 1 1 3 3 4
Output:
1 0 0 0 0 1 1 1 1 1

Test Case 2 Input:
5
5
2 4 2 4 1
Output:
2 0 2 0 0