CSCE 626 - Parallel Algorithm Design and Analysis

Programming Assignment 1

Name: Peihong Guo **UIN**: 421003404

1 Introduction

In this assignment, two versions of prefix sums are implemented, one using OpenMP and the other using MPI. The algorithms in both versions are the same except for API specific details.

2 Implementation Details

2.1 Algorithm

The algorithm implemented in this assignment is one based on local prefix sums: under the assumption that the number of processors p is much smaller than the number of integers n in the input sequence, each processor is assigned a chunk of integers and perform compute local prefix sums for the chunk. The processors then communicate with each other to compute a prefix sums for the partial sums and use this information to update local prefix sums to obtain the final results. The pseudo code below summaries the algorithm.

```
generate input sequence;
distribute the input sequence to each processor;
partial_sum = array of size nprocs + 1, initialized by 0

for each processor at rank tid
    compute local prefix sum
    copy the last element of the local prefix sum to partial_sum[tid+1]

compute the prefix sum of partial_sum in processor 0, in place

for each processor at rank tid
    add partial_sum[tid] to local chunk

if enabled, copy the local prefix sums back to processor 0
```

2.2 Differences between two versions

OpenMP version In this version, the distribution of input sequence is done implicitly because the input sequence is stored in a shared array. Each processor can access its own chunk of data directly. The local prefix sum is computed in place in this version for performance consideration. Other than the above mentioned places, the OpenMP version is the same as described in the pseudo code in previous section.

MPI version The MPI version is in fact a straight forward translation of the pseudo code. One thing worth noting is that explicit data distribution and collection through messages is used in MPI version because of the used API.

2.3 Miscellaneous

Both versions of prefix sum accepts an arbitrary number of integers (any number between 0 and `INT_MAX` ($2^{31} - 1$)), as well as arbitrary number of processors (although this is limited by the hardware).

To avoid the integer overflow, both versions uses ONLY `long` types for storing integers. In 64-bit system, `long` type can hold values between $-2^{63} + 1$ and $2^{63} - 1$. As long as the number of integers does not excceed `INT_MAX`, the prefix sum will not overflow.

At the end of the program, a verification step is added to check if the prefix sum computed in parallel is correct. This is done by comparing the parallel result against the result by a sequential algorithm.