Correlation Coefficient

Abby Malkey

July 11, 2014

1 Introduction

STAPL (Standard Template Adaptive Parallel Library) is the parallel version of the C++ STL, created in Parasol Lab at Texas A&M University. Parallel computing involves distributing the work required to run a program over several processors to decrease the run time. Test cases are created to evaluate the performance of basic algorithms and demonstrate their usability. This particular test case calculates the average time of an algorithm that finds the correlation coefficient of 2 vectors of floating-point numbers.

2 Design

The general design of the code is as follows:

```
create and fill vector/view (parse command line arguments for vector size
and filename)
calculate correlation coefficient by:
multiplying the size of the vectors by their inner product
summing each vector and multiplying the sums
subtracting from the inner product result
squaring each element in the vectors
taking the square root of the square of the elements and subtracting the
sum of that respective vector squared
dividing everything into the inner product result
output results
```

Algorithm 1: Correlation Coefficient

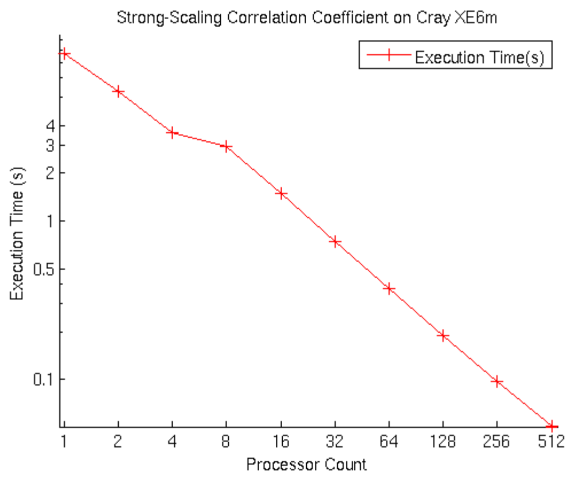
3 STAPL Components

The following parts of the STAPL library were used:
vector/vector view: store elements of data set to be processed
accumulate: sums all the elements of the vectors
stream: wrapper for an STL ifstream that is used to create an input stream to

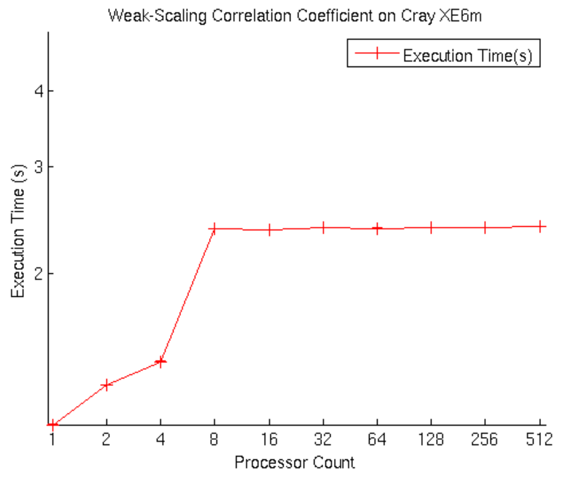
read from a file
serial_io: successively applies input operations to elements of the input view on location 0
map_reduce: applies multiple work functions to the elements of multiple views and reduces to a single answer to be assigned to a variable
counter: creates a counter to record the time
do_once: performs action on only one location
inner_product: used to calculate the inner (dot) product of two vectors and adds the result to the existing result
plus(): return the sum of its arguments, used for calculating the correlation coefficent

4 Results

The experiments were run on the Rain system, a Cray XE6m supercomputer with 576 cores at Texas A&M. Strong-scaling and weak-scaling trials were performed. The difference between strong and weak scalabilty refers to the amount of work the processor must perform. Strong-scaling is when a constant amount of work is loaded into an increasing number of processors, so each processor has an increasingly smaller workload. Weak-scaling invloves increasing the workload and the processor count at the same rate. The graphs below show the mean execution times of 32 samples with respect to the core count and 95 percent confidence intervals.



(a) Strong-scaling



(b) Weak-scaling

5 Analysis

The anticipated outcomes were for the times to decrease dramatically for strong-scaling and hold steady for weak-scaling. For strong-scaling, the execution time drops quickly as the processor count is increased since the work is being split among the processors. The weak-scaling results were as predicted, varying only slightly, except for a slight jump when going from 4 to 8 cores. This is due to the memory bandwidth being completely utilized, as all 8 cores were placed by the batch system onto a single die that has limited bandwidth to the memory on the compute node. The confidence intervals were small enough to suggest that the mean times were well within reason.

6 Conclusion

In conclusion, simple statistics algorithms can easily be written in STAPL, which speeds up the execution time by distributing the work over a chosen number of processors.