Christopher Smith
Parallel program 2
7 November 2014

Submition

makefile – compiles code
SatSource.c – circuit-satisfiability problem timed with static and dynamic scheduling
find_primes.c – finds the primes up to a given n, with timed serial, static, and dynamic scheduling

Description of the program:

The file SatSource.c is a parallel program on the circuit-satisfiability problem. This program splits the 65,536 possible combinations of inputs into threads and prints all the results that would output a one. The program is timed using omp_get_wtime() and both dynamic and static scheduling are used and compared.

The file find_primes.c is a parallel program that finds all the primes between 2 and some n value entered on the command line with the executable. It is a parallel version of the sieve of Eratosthenes with the outer loop being parallel and the inner loop that finds the multiples of the primes is not. The program is deterministic if multiple threads reach the same multiple of their corresponding prime number and try writing false to the spot in the array. This program times the serial version and also to scheduling types of static and dynamic.

Compiling and Running program:

Compile:
make

Run:
./SatSource (number of threads)
./prime (number to find primes up to)

clean:
make clean

Testing and Verification:

SatSource was tested to make sure no race conditions were happening since the function was given.

Find primes was tested on low values of n to verify that it was finding all the primes and only the primes. I did implement a counter so that I only had to see how many it was finding for larger values of n greater than a million so that the number of primes found could be compaired to the number found in online sources.

Timing Results

SatSource was ran with timed versions of serial, static scheduling, and dynamic scheduling.

Dynamic tended to be faster than both static and serial when thread count was equal to eight. When the thread count was lower the timing was slightly slower that statics time but still faster than serial. This is because static schedule type can have its blocks computed ahead of time and distributed evenly between therrads. While dynamic scheduling is done at run-time and has more overhead than static so is slower with less threads.

Finding the primes and the timings of how long it took between serial, dynamic, and static show similar results to SatSource. On lower values of n of less than a million static typically ran faster than both serial and dynamic. As n got greater than a million Dynamic tended to surpass both static and serial by a fairly large margin. With 10000000 numbers to sort through and find the prime numbers:

Serial Time: 0.155558 Number of Primes: 664578
Dynamic Time: 0.083861 Number of Primes: 664578
Static Time: 0.098469 Number of Primes: 664578

On 100000000
Serial Time: 1.333610 Number of Primes: 5761454
Dynamic Time: 0.878520 Number of Primes: 5761454
Static Time: 1.183023 Number of Primes: 5761454

The obove two timings show that the Dynamic time is consistently faster as n gets bigger and that static is only slightyly faster than serial.