Project 2: Parallel Fluid Dynamics Simulation

John Bradley

April 11, 2024

1 Introduction

This report aims to document the implementation and analysis of a program that utilizes OpenMP to implement a computational fluid dynamics model suitable for modeling behavior of turbulent flows. This report will cover the methods and techniques utilized for implementing the fluid dynamics model using OpenMP, analyze and estimate the expected performance of the implementation, present the results from experimentation, compare and contrast the expected performance to the observed results, and conclude with insights and possible changes that could be made to the implementation

2 Methods and Techniques

Many of the functions in the code provided deal with sequential sections of memory, and allowing a single thread to access these sequential memory areas would be beneficial. Additionally, there are many nested loops. The Ptolemy system does not have `OMP_NESTED` set by default, thus the use of nested `#pragma omp for` declarations should be avoided. To do so, careful selection of partitioning of parallel and serial sections need to be made to exploit sequential memory access.

For example, in the `setInitialCOnditions()` function, the inner-most nested loop contains:

Listing 1: Example C code

```c
for(int k=0; k<nk; ++k) {
  int indx = offset + k;
  float dz = (1./nk)*L;
  float z = 0.5*dz+k*dz - 0.5*L;

  // 3-D taylor green vortex
  u[indx] = 1.*coef*sin(x/l)*cos(y/l)*cos(z/l);
  v[indx] = -1.*coef*cos(x/l)*sin(y/l)*cos(z/l);
  p[indx] = (1./16.)*coef*coef*(cos(2.*x/l)+cos(2.*y/l))*(cos(2.*z/l)+2.);
  w[indx] = 0;
}
```

For each iteration of the loop, `indx` is the sum of `offset`, defined in the outer loop, and the value of `k`, which is incremented each iteration of the loop. If the loop was parallelized, each thread may access non-sequential sections of the `u`, `v`, `p`, and `w` arrays. If we, instead, parallelize the loop that contains the "`k`" loop, then we can exploit this memory access pattern

Similarily, this pattern of access is present across many of the functions in the program, and can be exploited in the same way.

3 Analysis

4 Results

5 Synthesis

6 Conclusion