**OpenMP Parallel Programming Project Report**

Rishik Sarkar (rs1894)

Experimental Results Table:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **Avg.** | **Med.** | **Std.** |
| **s-2** | 145.23 | 145.19 | 144.64 | 145.85 | 145.71 | 144.23 | 144.84 | 148.31 | 143.90 | 143.05 | **145.10** | **145.02** | **1.339** |
| **sol** | 146.18 | 145.11 | 143.55 | 144.77 | 149.18 | 146.83 | 188.94 | 146.19 | 142.42 | 144.60 | **149.78** | **145.65** | **13.172** |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **Avg.** | **Med.** | **Std.** |
| **f-2** | 137.84 | 136.91 | 137.06 | 132.02 | 134.43 | 135.61 | 136.47 | 137.19 | 136.90 | 139.05 | **136.35** | **136.91** | **1.853** |
| **sol** | 137.23 | 137.69 | 137.42 | 156.07 | 153.51 | 138.24 | 140.93 | 137.91 | 136.06 | 139.86 | **141.49** | **138.08** | **6.797** |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **Avg.** | **Med.** | **Std.** |
| **s-4** | 69.29 | 68.20 | 68.99 | 68.06 | 67.88 | 72.47 | 68.90 | 68.86 | 68.39 | 67.45 | **68.85** | **68.63** | **1.322** |
| **sol** | 68.58 | 68.73 | 70.33 | 67.96 | 69.53 | 68.11 | 67.68 | 70.50 | 73.45 | 67.69 | **69.26** | **68.66** | **1.704** |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **Avg.** | **Med.** | **Std.** |
| **f-4** | 69.06 | 69.59 | 67.80 | 67.66 | 69.40 | 69.62 | 69.02 | 66.66 | 69.45 | 68.71 | **68.70** | **69.04** | **0.947** |
| **sol** | 68.96 | 69.57 | 68.13 | 68.62 | 68.48 | 69.50 | 69.28 | 69.01 | 72.59 | 69.13 | **69.33** | **69.07** | **1.169** |

Notes:

* The **rows** represent particular **code versions** with their respective solution file right below. The file names are as follows:
  + **s-2**: *spell\_t2\_singleloop*
  + **f-2**: *spell\_t2\_fastest*
  + **s-4**: *spell\_t4\_singleloop*
  + **f-4**: *spell\_t4\_fastest*
* The first ten **columns** represent the **execution times (in ms)** of each experiment conducted on the specified code version of the row. The last three columns represent the **average (mean), median, and standard deviation** of the entries of all 10 experiments in each respective row.
* The execution times have been rounded to 2 decimals (except standard deviation, which is rounded to 3 decimals) in order to fit the data within the table
* All the experiments were conducted on **iLab1.cs.rutgers.edu**, and attempted to spell check the word “**principles**”

Parallelization Strategy:

* I chose to parallelize the **outer loop** in the nested for-loop that applies every hash function in hf to every word in the dictionary of words and fills the bit vector (bv). I picked this loop because:
  + Each iteration is independent, so we can distribute them over multiple threads
  + Parallelizing the outer loop is better in this case since it seems to have a lower overhead than the inner loop
* For most of the programs, I used the following statement:

**#pragma omp parallel for private(i, j, hash) schedule(guided) shared(hf, bv\_size, bv)**

* + Parallelizes the outer for loop
  + Creates private copies of the local variables i, j, and hash in each thread
  + Divides the iterations into similar sized chunks that decrease in size later in order to provide optimal load balancing. I mainly used this because it led to much better performance than static or dynamic scheduling
  + Shares the variables/data structures hf, bv\_size, and bv across all threads