Livermore loops optimization with OpenMP

**Kernel 21:**

* optimization manner:
  + changing loops order (bigger loop be outer)
  + loop unrolling (5 times)
  + omp\_set\_num\_threads(THEADS);
  + #pragma omp parallel shared(px, vy, cx) private(k, i, j)
* compiling:

gcc -fopenmp ker21.c -m64 -lrt -lc -lm -o ker21

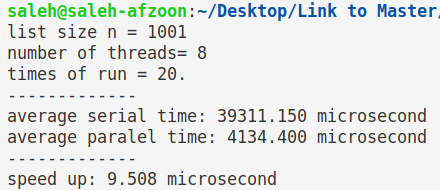
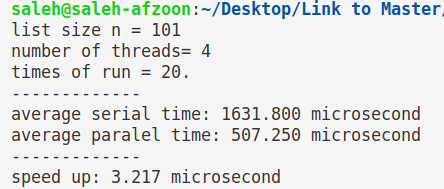
* analyzing best number of threads:

average speedup per 20 time of run

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Threads** | **2** | **3** | **4** | **5** |
| **N = 101 (array size)** | 1.780 | 2.908 | 3.217 | 2.264 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Threads** | **6** | **8** | **10** | **12** |
| **N = 1001 (array size)** | 7.373 | 9.508 | 0.862 | 0.782 |

* results:

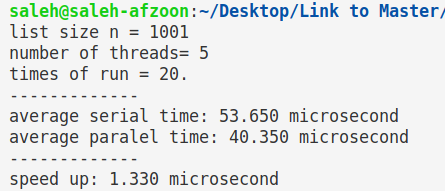
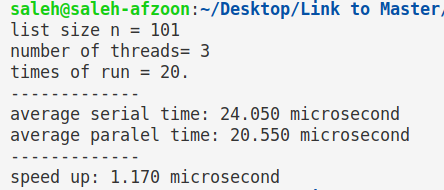


**Kernel 22:**

* optimization manner:
  + loop unrolling (4 times)
  + omp\_set\_num\_threads(THEADS);
  + #pragma omp parallel shared(y, u, v, w, x) private(k)
* compiling:

gcc -fopenmp ker22.c -m64 -lrt -lc -lm -o ker22

* results:

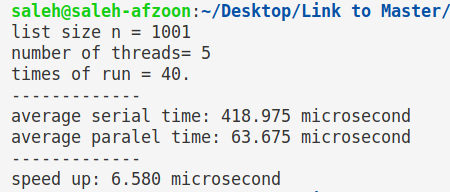
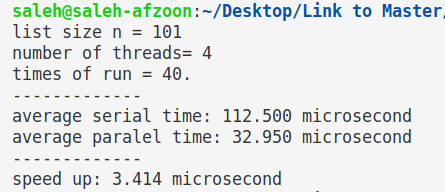


**Kernel 23:**

* optimization manner:
  + changing loops order (bigger loop be outer)
  + loop unrolling (2 times)
  + omp\_set\_num\_threads(THEADS);
  + #pragma omp parallel private(j)
  + collapsing nested loops with #pragma omp collapse(2)
* compiling:

gcc -fopenmp ker23.c -m64 -lrt -lc -lm -o ker23

* results:

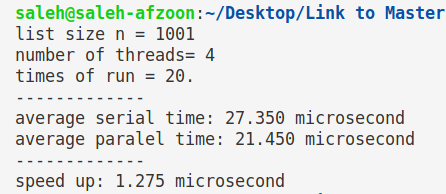


**Kernel 24:**

* optimization manner:
  + loop unrolling (4 times)
  + omp\_set\_num\_threads(THEADS);
  + #pragma omp parallel private(k)
* compiling:

gcc -fopenmp ker24.c -m64 -lrt -lc -lm -o ker24

* results:



**Kernel 24:**