Livermore loops optimization with OpenMP

* **Kernel 21:**
  + non optimized:

for (k = 0; k < 25; k++){

for (i = 0; i < 25; i++){

for (j = 0; j < n; j++){

px[j][i] += vy[k][i] \* cx[j][k];

}

}

}

* + optimization manner:

at the first loops order changed in manner that the bigger loop (last inner loop) be outer loop in order to doing the parallelism on the bigger area.

second loop unrolling performed on the on the new inner loop

and at the end pragma statement add in the way that specifies px,vy,cx as shared resources and the

* + compiling:

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

* + result:

kernel 22 , 1001 => 4 threads

kernel 23 => 4 threads

**Q2**

* Approach:

The main PI calculation loop was divided based on number of threads and gave to threads for processing. An array of results created to store results of threads; each thread writes its own result in a cell of array.

In the end sum of all results used to calculate value of PI.

The information about each thread process gave to them as bellow structure:

double \*\*sums = (double \*\*)malloc(sizeof(double\*) \* threads\_num);

typedef struct argument{

int id;

int size;

int threads\_num;

double \*sum;

}argument;

* Commands:

Compile command:

gcc -IC:/MinGW/include/ -pg pi.c -lpthread -o pi.exe

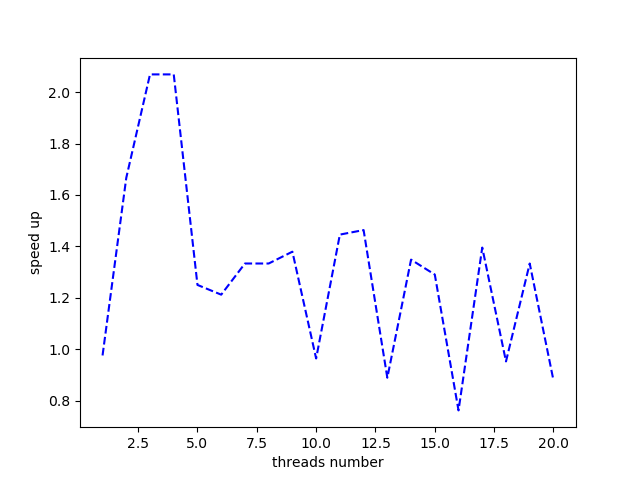
pi.exe <max threads num>

the input is max thread number. Code runs the process 40 time per each number of threads and return average PI, time and speed up value and store all speed ups into file “speedups.txt”.

plot.py

then by use of ***pyplot*** content of “speedups.txt” file used as input to plot speed up diagram.

* Comparison:



As the plot shows by increasing threads number, in small number of threads we have actual speed up and in the 2 or 3 number of threads is the best. It seems that from a point till end by adding more threads, the overhead of managing threads make the process slower than serial manner and doesn’t gave any benefits cause the main algorithm isn’t really complex and time consuming and 2 or 3 threads is enough.