CS4552 - Scientific Computing

Assignment 1 - Matrix Multiplication Performance with Loop Ordering

Consider a multiplication of two matrices, each with N x N dimensions. The matrix multiplication function involves three loops. Example:

```
void multMat1( int N, int *A, int *B, int *C ) {
    int i,j,k;
    /* This is ijk loop order. */
    for( i = 0; i < N; i++ )
        for( j = 0; j < N; j++ )
            for( k = 0; k < N; k++ )
                C[i+j*N] += A[i+k*N]*B[k+j*N];
}
```

However, the order of the loop nesting does not impact the final result. There are six choices you can consider for organizing the loop nesting as follows:

(1) ijk, (2) ikj (3) jik (4) jki (5) kij (6) kji

Notice the above example code is arranged as ijk loop order. Assume that matrices A, B, and C are all N-by-N and stored in one-dimensional column-major arrays.

Use C language with gcc compiler to complete the following tasks. Use -O0 no optimization

Task 1:

Implement the matrix multiplication functions for floating point numbers and integers, all six loop ordering choices.

Task 2:

Implement a function to record the starting time and end time of matrix multiplication tasks. Report the time consumed by the matrix multiplication function.

Task 3:

Allocate memory for three NxN matrices, where N can be given as a command line argument to the executable file. Two matrices are for inputs, (A and B) and the third matrix (C) will be used for storing the result. Populate the two input matrices with pseudo random numbers with the current time as the seed. You can also seed with a constant for repeatability.

Use valgrind tool to verify there is no memory leaks or other memory access errors.If there are any errors, make sure to fix them before proceeding further.

valgrind --leak-check=yes --log-file=valgrind.rpt ./a.out

Task 4:

Write the code to multiply A and B, storing the result to C. Compile the code with gcc using zero optimization level (-O0) and include the debug information (-g).

```
gcc -g -O0  matmul1.c
```

Measure the time consumed by matrix multiplication time for all alternative loop orders. If you repeat the exact same experiment multiple times, what is the variance of the measured time consumption? Justify and give reasons for any observed variance.

Task 5:

Write a shell script to measure and record the time consumed by each method when the matrix dimensions N is changed. Example: N = 10, 100, 1000, 10000 and any other suitable values for N. Be careful not to include the time for printing matrices or any other additional instructions, but just the matrix multiplication instruction. Report the available main memory and estimated memory utilization during the execution of the program. Plot the results. Is there a significant difference of execution time between the different loop orders?

Task 6:

Which 2 two loop-orders would perform best for these 1000-by-1000 matrices? Which 2 loop-orders would perform the worst? Describe your observations.

Task 7:

Use lscpu command to identify the available cpu instruction set extensions. Report the output.

Task 8:

What are the other alternatives you could take to further reduce the runtime of the program? (Example: -O3 enabled) Describe further optimizations and record the time. Plot the results along with the best-case loop ordering scenario.

You can use the following development server for executing the experiments. It can be accessed only from within the university network. If you need remote access, request VPN access via https://helpdesk.uom.lk/open.php Then install and establish the VPN before logging in.

- Server: 10.8.100.115
- Username: last three numbers and the letter in your index number.
- Password: cs4552test (reset your own password at the first login)
- Please be considerate about the other users of the system. Put a reservation in this google sheet , especially if you are executing final data collection.

- Observe the current utilization of resources in the machine using htop utility before starting a new execution to make sure sufficient resources are available.