Operating Systems CSE x61 lab 2 – multi-threading

es-amin.mohamedamin2026@alexu.edu.eg

Amin Mohamed Amin El-Sayed Amin

21010310

- **Global Variables:**
- Matrices for input and output data.
- File pointers for input and output files.
- Variables for storing dimensions of input matrices.
- Variable to check if input is the default case.

- Functions Description:

- **read_matrices_from_file(FILE* fptr, int is_matrix_a):** Reads input matrices from files, extracting dimensions and matrix elements based on the provided file pointer and flag indicating the matrix type (A or B).

- **write_matrices_in_file(int matrix[MAX][MAX], FILE *fptr):** Writes output matrices to files, including dimensions followed by matrix elements, using the provided matrix array and file pointer.

- **thread_per_entire_matrix():** Performs matrix multiplication with one thread for the entire matrix, computing each element by multiplying corresponding rows and columns of input matrices.

- **thread_per_row(void *row_index):** Performs matrix multiplication with one thread per row, calculating each row of the output matrix independently using the specified row index.

- **thread_per_element(void *element)**: Performs matrix multiplication with one thread per element, computing each element of the output matrix independently based on provided row and column indices.

- **construct_thread_per_entire_matrix()**: Manages matrix multiplication with one thread per entire matrix, handling timing, thread creation, and execution.

- **construct_thread_per_row():** Manages matrix multiplication with one thread per row, handling timing, thread creation, and execution.

- **construct_thread_per_element():** Manages matrix multiplication with one thread per element, handling timing, thread creation, and execution.

- **initialize_files(char* names[]):** Initializes file pointers based on input arguments, opening input and output files for reading and writing.

- **main(int argC, char* args[]):** Entry point of the program, reads input matrices, performs matrix multiplication using different threading methods, and writes output matrices to files based on command-line arguments.

Compilation and Execution:

- To compile and run the code, follow these steps:
- Compile the code using a C compiler (e.g., gcc):
- `gcc main.c -o main -pthread`

Run the compiled program:

- `./main <input_file_a> <input_file_b> <output_file_prefix>`
  - **`<input_file_a>`**: Path to the file containing the first input matrix.
  - **`<input_file_b>`**: Path to the file containing the second input matrix.
  - **`<output_file_prefix>`**: Prefix for the output files.

- The performance of each method (thread per matrix, thread per row, and thread per element) depends on various factors such as the size of the matrices, hardware architecture, and overhead associated with thread creation and synchronization. In some cases, the first method (a thread per matrix) might outperform the other methods due to the following reasons:
- **Less Thread Overhead:** Creating fewer threads (only one thread for the entire matrix) reduces the overhead associated with thread creation, context switching, and synchronization compared to creating multiple threads per row or per element.
- **Reduced Synchronization Overhead:** With only one thread, there's no need for explicit synchronization mechanisms such as mutexes which can introduce overhead when coordinating multiple threads accessing shared resources.

- For test cases click on the following link
- **https://github.com/WorldisAmen/Matrices-Multiplication-Multi-Threading**