Here is the speedup graph, where we compare the speedup achieved with different numbers of CPUs for varying matrix sizes. Each line represents a different matrix size and is color-coded for distinction. The x-axis is on a logarithmic scale to better display the range of CPU counts, and the y-axis is also logarithmic to clearly show the speedup values, especially since they span several orders of magnitude.

From this graph, we can see how the speedup increases with the number of CPUs. However, the rate of increase varies depending on the matrix size, with larger matrices generally benefiting more from additional CPUs. This is expected due to the higher computational load of larger matrices, which provides more opportunity for parallel processing to improve performance

the speedup trends for matrix sizes 6 and 36 seem unexpected compared to the larger matrix size of 1296. This discrepancy could be due to several factors:

1. **Problem Size**: For smaller matrices (6 and 36), the overhead of parallelization (e.g., thread creation, synchronization) might outweigh the benefits gained from parallel execution. In other words, the computational workload might not be large enough to fully utilize all available CPU resources efficiently.
2. **Communication Overhead**: In some cases, especially with smaller problem sizes, the overhead of communication between CPUs can become significant compared to the actual computation time. This overhead can limit the speedup achievable with parallelization.
3. **Parallel Efficiency**: The parallel efficiency, which measures how effectively additional CPUs contribute to speeding up the computation, might be lower for smaller problem sizes. This could be due to factors such as load imbalance among CPUs or contention for shared resources.
4. **Algorithmic Complexity**: The algorithm used for matrix operations might have different scaling characteristics depending on the matrix size. Some algorithms might exhibit better parallel scalability for larger matrices compared to smaller ones.

To further investigate these discrepancies, it might be helpful to analyze the specific characteristics of the computation, such as the parallelization strategy used, the communication patterns, and the computational complexity of the algorithm. Additionally, profiling the code execution and analyzing the behavior of the system under different configurations could provide insights into the observed performance differences.