E0 - 243 Computer Architecture Assignment - 2

Mandar Bhalerao - mandarmanoj@iisc.ac.in
Kunal Pansare - kunaldinesh@iisc.ac.in

November 26, 2023

1 Introduction

We are required to optimize the problem of "Dilated Convolution(DC)" which involves an input matrix and a kernel matrix. The output of the dilated convolution is stored in the output matrix. The dimensions of these matrix are:-
Input:
1. Input Matrix: Input_Row x Input_Column
2. Kernel Matrix: Kernel_Row x Kernel_Column

Output:
Output Matrix: (Input_Row-Kernel_Row+1) x (Input_Column-Kernel_Column+1)
We will Optimize the Single Threaded Dilated Convolution and Implement and Optimize Multi-Threaded Dilated Convolution of the given unoptimized implementation provided on GitHub.

2 Optimization Methods used and Performance Analysis

2.1 Optimization Methods

I.Loop Unrolling:- Loop unrolling is a compiler optimization technique that aims to improve the performance of a program by reducing the overhead of loop control code. The idea is to process multiple iterations of a loop in a single iteration, effectively reducing the number of loop-control instructions and improving instruction-level parallelism.

II.Elimination of Redundant Computations:- Redundant computations refer to performing the same calculation multiple times when the result remains unchanged. Some strategies used are:-
a)Storing Intermediate Results
b)Loop Invariant Code Motion
c)Common Subexpression Elimination
d)Use of Temporary Variables

III.Strength Reduction:- Strength reduction is a compiler optimization technique that involves replacing expensive operations with equivalent, but less expensive, operations. The goal is to improve the overall performance of a program by reducing the computational cost of certain expressions.

IV.Single Instruction, Multiple Data(SIMD):- SIMD stands for Single Instruction, Multiple Data. It's a type of parallel computing architecture that performs the same operation on multiple data elements simultaneously. In simpler terms, SIMD allows a processor to apply a single instruction to multiple pieces of data at the same time, improving the overall efficiency of computations.

2.2 Performance Analysis

Given below is the Performance Analysis of obtaining the output matrix when it is executed for combinations of input matrix dimensions and kernel matrix dimensions respectively. The Performance Metrics used for analysis are:-

[PartA-I] Optimize single-threaded DC(CPU)

a. Improvement in Single Threded Execution(%) :- The metric calculates the percentage reduction in execution time achieved by the single-threaded implementation compared to a reference execution. This metric quantifies the efficiency gain achieved by the single-threaded approach, expressing it as a percentage improvement relative to the reference execution time.

Improvement(%)=(1-(Single Thread Execution Time/Reference Execution Time))*100

b. Speedup in Single Threded Execution :- Speedup is a performance metric that quantifies the improvement in the execution time of a program or algorithm when compared to a reference or baseline execution. It is typically expressed as the ratio of the reference execution time to the improved execution time. The speedup is calculated using the formula:

Speedup=Reference Execution Time/Single Thread Execution Time

Sr. No	Kernel Matrix dimensions	Reference Execution Time(ms)	Single Thread Execution Time(ms)	Imp in STE(%)	Speedup(STE)
1	3x3	925.2	364.7	60.57	2.53
2	5x5	2427.8	443.1	81.74	5.47
3	7x7	4705.6	795.7	83.08	5.91
4	9x9	7779.0	1253.3	83.88	6.20
5	11x11	11586.3	1848.6	84.0	6.26
6	13x13	16511.1	2517.71	84.75	6.55
7	64x64	352101.0	63791.7	81.88	5.51

Table 1: Input Matrix with dimensions 4096x4096

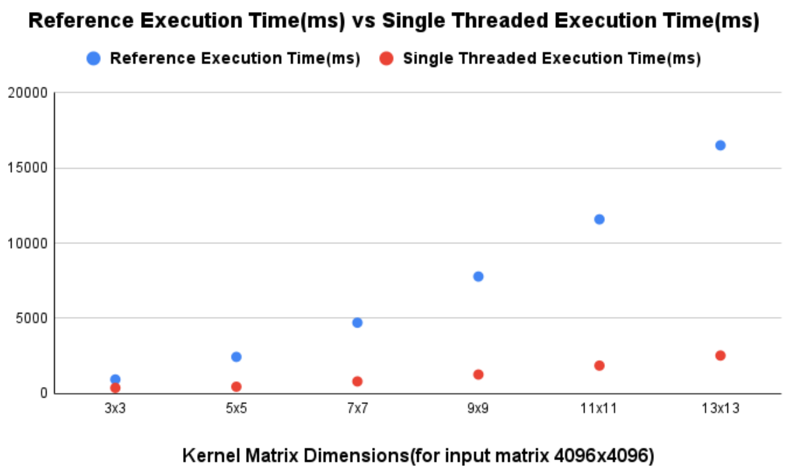

Figure 1

Sr. No	Kernel Matrix dimensions	Reference Execution Time(ms)	Single Thread Execution Time(ms)	Imp in STE(%)	Speedup(STE)
1	3x3	3711.4	1489.8	59.85	2.49
2	5x5	10019	1767.7	82.35	5.66
3	7x7	19569.1	3275.9	83.25	5.97
4	9x9	31386.6	5043.3	83.93	6.22
5	11x11	46482.1	7350	84.18	6.32
6	13x13	66888.1	10088.4	84.91	6.63
7	64x64	1431110	247423	82.71	5.78

Table 2: Input Matrix with dimensions 8192x8192

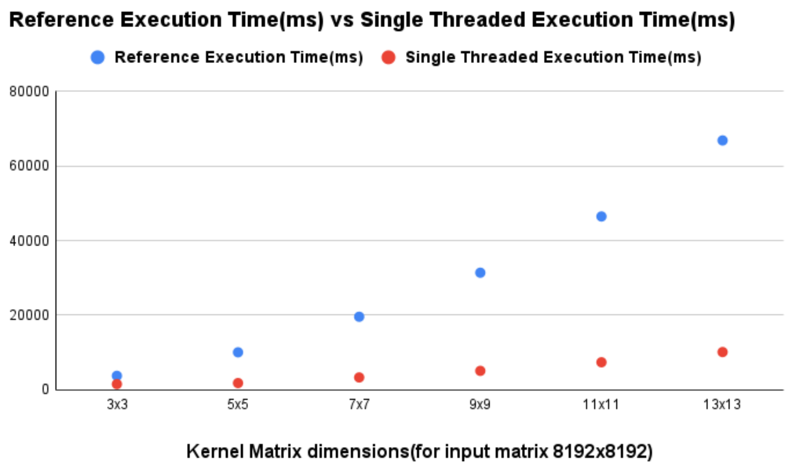

Figure 2

Sr. No	Kernel Matrix dimensions	Reference Execution Time(ms)	Single Thread Execution Time(ms)	Imp in STE(%)	Speedup(STE)
1	3x3	14921.3	5886.6	60.54	2.53
2	5x5	40086.5	7028.3	82.46	5.70
3	7x7	77150.9	12828.1	83.37	6.01
4	9x9	125654	19902.6	84.16	6.31
5	11x11	186615	28219	84.87	6.61
6	13x13	267618	38078	85.77	7.02

Table 3: Input Matrix with dimensions 16384x16384

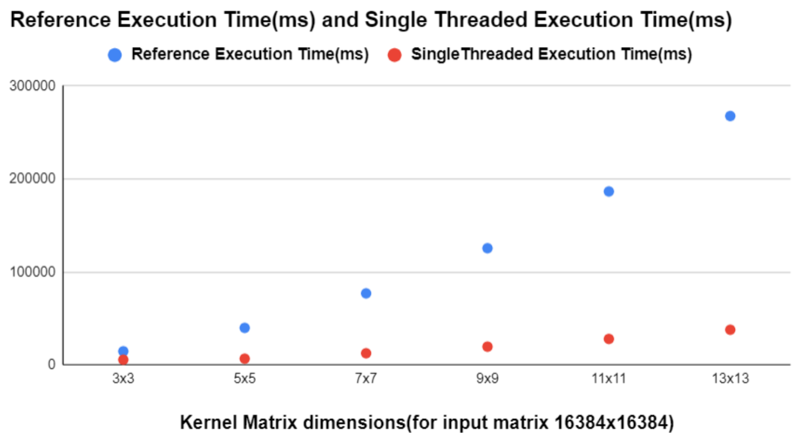

Figure 3

2.3 Analysis using perf tools

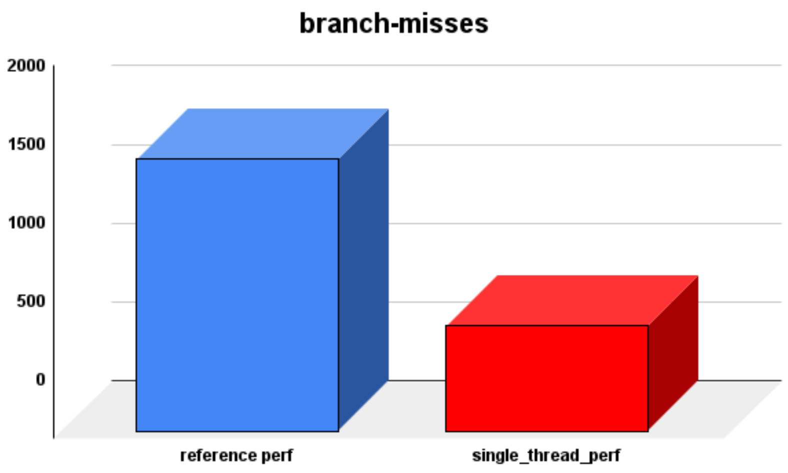

Figure 4

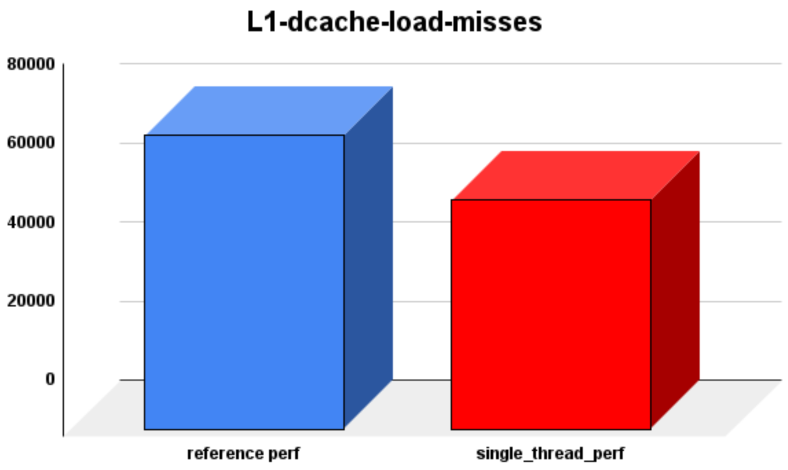

Figure 5

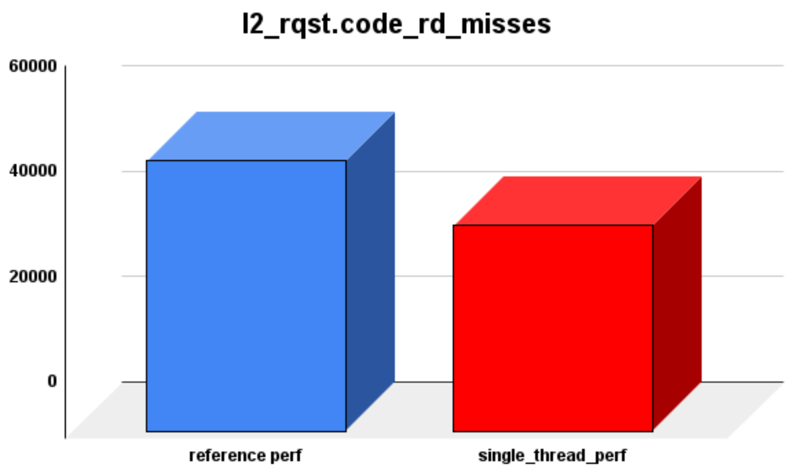
l2_rqst.code_rd_misses
60000
40000
20000
0
reference perf
single_thread_perf

Figure 6

[PartA-II] Optimize multi-threaded DC(CPU)

It refers to implementing dilated convolution in a multi-threaded programming paradigm. Multi-threading involves dividing a program into smaller threads that can run concurrently, potentially improving performance on multi-core processors.

Sr. No	Kernel Matrix dimensions	Reference Execution Time(ms)	Multi Thread Execution Time(ms)	Imp in MTE(%)	Speedup(MTE)
1	3x3	925.2	364.7	60.57	2.53
2	5x5	2427.8	443.1	81.74	5.47
3	7x7	4705.6	795.7	83.08	5.91
4	9x9	7779.0	1253.3	83.88	6.20
5	11x11	11586.3	1848.6	84.0	6.26
6	13x13	16511.1	2517.71	84.75	6.55
7	64x64	352101.0	63791.7	81.88	5.51

Table 4: Input Matrix with dimensions 4096x4096

Number of Threads / Input Matrix Dim.	Reference(ms)	4	8	12	16
1024	998.1	48.27	46.49	39.85	47.65
2048	4039.8	356.16	339.31	347.46	281.78
4096	16511	1679.7	1754.2	1850	1830
8192	66888.1	6590.42	6906.5	7530	7490.9

Table 5: Kernel Matrix with dimensions 13x13

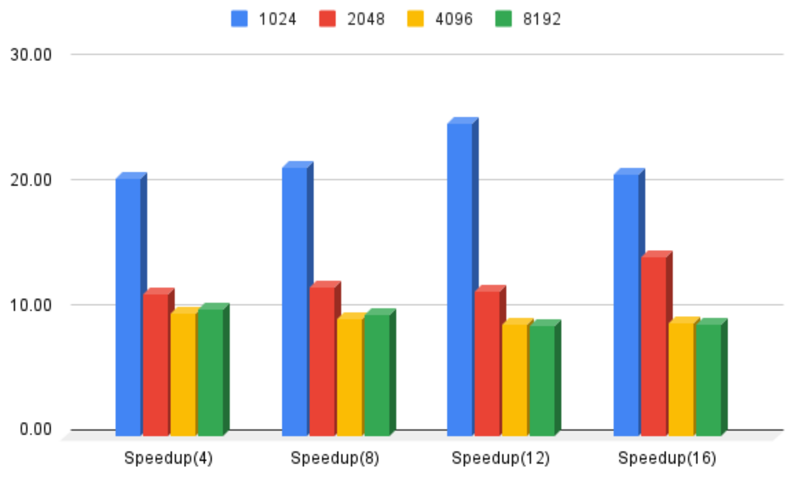

Figure 7

Number of Threads / Input Matrix Dim.	Reference(ms)	4	8	12	16
1024	20996	1595.4	1445.3	1312.2	1379.7
2048	90530.2	8682.64	7066.9	7033	6159.3
4096	352101	40590.5	40415.4	44033	43298.3
8192	1431110	157475	165831	176217	175417

Table 6: Kernel Matrix with dimensions 64x64

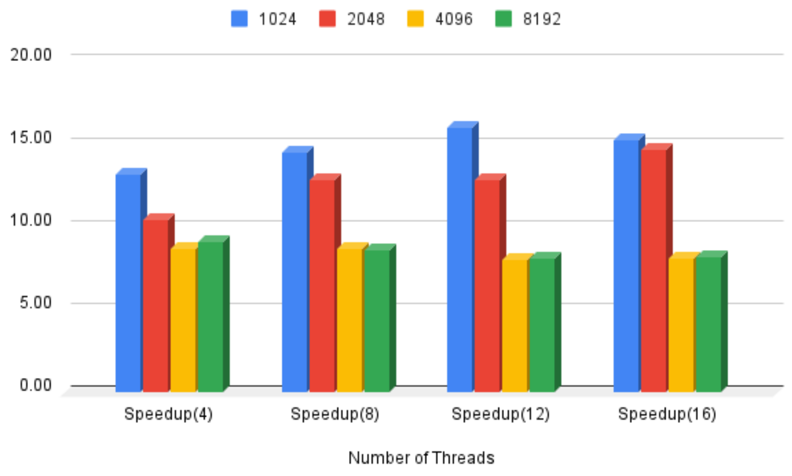

Figure 8

3 Conclusion

3.1 Conclusion for Optimized Single-Threaded Dilated Convolution

In conclusion, the optimized single-threaded dilated convolution implementation has demonstrated notable improvements in performance. By strategically incorporating loop unrolling, memory access optimizations, Elimination of redundant computations, SIMD, Strength Reduction we have significantly reduced the execution time. This optimized version maintains the accuracy of the convolution operation while leveraging computational resources more efficiently on a single processing core.

3.2 Conclusion for Multi-Threaded Dilated Convolution

The multi-threaded dilated convolution implementation has demonstrated the power of parallelism in accelerating the computation of dilated convolutions. By dividing the workload across multiple threads, we harnessed the capabilities of modern multi-core processors to achieve substantial reductions in execution time.