International
Journal
Of Advanced Research in Engineering & Management (IJAREM)

Optimizing the use of multi cores by image processing application in desktop

M. Pavithra, G. Abinaya
MS Software Engineering VIT University, Vellore.

Abstract *:* now a days, we see multi core processors everywhere, a customer who uses a multi core processor expects it to be that many times faster than a single core processor but do those computers actually give that expected output is an unanswered question, hence in this paper we are planning to optimize an existing code in such a way that it utilizes those multiple cores efficiently. The application that we are going to optimize in this paper is image processing application which is one of the most used applications by today's desktop users. Our paper will efficiently bring out the processor utilization by the application chosen for the paper.
Keywords: , Multicore, Parallelization, Quantization.

I. INTRODUCTION

Today due to the need for faster performance multi core processors became very common. Yet the full utilization of them is not done efficiently, in this paper we provide a measurement and analysis of the existing (We have chosen Image compression application) and optimized the code in terms of speed and compression factor, here we optimize the application, so that the same functions could be performed as before, but with a faster execution time. Thus the hidden capability of the multicore processor environment could be utilized well.

II. EXPERIMENTS PERFORMED AND RESULTS OBTAINED

A. Experiment

In this paper we are going to take one of the most commonly used desktop application, and optimize its code so that it executes faster. We have chosen Image compression application, as it is one of the most commonly used desktop application today.

And we have selected three image compression algorithms, two based on the use of a frequency transform such as DCT and one based on the use of a Wavelet transform and performed their scalar execution and tabulated (Table 1) their results to choose the best algorithm out of the selected algorithms, there are many parameters to compare their efficiency we have used the execution speed, image size, compression factors as our parameters for comparison.

Table 1: Comparison of three algorithms

S.No	Image Name	Image Format	Original Size	Algorithm 1		Algorithm 1		Algorithm2		Algorithm2		Algorithm3		Algorithm3	
				Speed	Size	Compression ratio		Speed	Size	Compression ratio		Speed	Size	Compression ratio	
1	Image1	jpeg	85.4 KB	5.125 sec	9.07 KB		0.115	0.983 sec	19.5 KB		0.228	8.457 sec	55.2 KB		0.646
2	Image2	jpeg	11.0 KB	1.278 sec	5.73 KB		0.52	0.217 sec	11.0 KB		1.072	3.961 sec	16.0 KB		1.454
3	Image3	png	278 KB	5.54 sec	16.8 KB		0.06	0.480 sec	28.4 KB		0.102	6.621 sec	37.6 KB		0.135
4	Image4	ico	60.2 KB	0.006 sec	not compressed	nil		0.063 sec	not compressed	nil		0.031 sec	not compressed	nil	
5	Image5	jpg	39.6 KB	2.135 sec	10.2 KB		0.257	0.312 sec	21.4 KB		0.54	4.221 sec	33.4 KB		0.843
6	Image6	bmp	4.06 MB	2.135 sec	109 KB		0.026	3.130 sec	110 KB		0.027	21.354 sec	292 KB		0.079
7	Image7	gif	59.3 KB	0.788 sec	not compressed	nil		0.032 sec	not compressed	nil		3.574 sec	156 KB		2.293
8	Image8	webP	57.7 KB	0.040 sec	not compressed	nil		0.057 sec	not compressed	nil		0.021 sec	not compressed	nil	
9	Image9	jpg	52.7 KB	3.216 sec	16.3 KB		0.309	0.499 sec	29.8 KB		0.565	3.963 sec	39.3 KB		0.745
10	Image10	jpeg	159 KB	13.213 sec	71.0 KB		0.446	0.810 sec	114 KB		0.716	9.665 sec	190 KB		1.194
11	Image11	jpeg	10.9 MB	1253 sec and still...	not compressed	nil		397 sec	not compressed	nil		matlab hangs..	not compressed	nil	

(Table 1)

Based on the results obtained, though algorithm2 has better speed, but the other two parameters are lesser, hence we chose algorithm1 and have vectorised it. Vectorizations is a processes in which multiple operations take place at a time, in our project We have vectorised the chosen algorithm by first splitting the read image matrix into two then we carried out the compression technique to each of these split matrix and we applied certain changes to the old code like preallocation of array size, parallelizing the inner loops and assignment of new variables. At the last we have combined the two split matrix into one to form the final compressed image output.
After vectorising we ran both the initial and the modified algorithm and calculated their execution times, the vectorised algorithm is found to run at twice the speed of the initial algorithm (Table 2). Thus a properly vectorised code that makes full use of the multicore could be faster by two times.

Table 2: Comparison of normal and vectorised code

S.No	Image Name	Image Format	Original Size	Algorithm 1	Algorithm 1	Algorithm 1	vecalgo	vecalgo	vecalgo
				Speed	Size	Compression ratio	Speed	Size	Compression ratio
1	Image1	jpeg	85.4 KB	2.541	11.0 KB	0.128	2.182	10.9 KB	0.127
2	Image2	jpeg	6.35 KB	0.762	5.71 KB	0.899	0.669	5.60 KB	0.881
3	Image3	png	278 KB	2.253	20.0 KB	0.719	1.775	19.8 KB	0.071
4	Image4	ico	60.2 KB	0.123	not compressed	nil	0.04	not compressed	nil
5	Image5	jpg	39.6 KB	0.021	15.8 KB	0.398	0.009	21.4 KB	0.54
6	Image6	bmp	4.06 MB	18.568	122 KB		8.133	122 KB	
7	Image7	gif	59.3 KB	0.859	not compressed	nil	0.05	not compressed	nil
8	Image8	webP	57.7 KB	0.064	not compressed	nil	0.03	not compressed	nil
9	Image9	jpg	52.7 KB	1.35	25.5 KB	0.483	1.298	25.6 KB	0.485
10	Image10	jpeg	159 KB	5.264	110 KB	0.691	3.145	110 KB	0.691
11	Image11	jpeg	10.9 MB	1253 sec and still..	not compressed	nil	397 sec	not compressed	nil
12	Image12	jpeg	12.8 KB	0.774	9.77 KB	0.763	0.742	9.74 KB	0.76
13	Image13	png	3.14 KB	0.232	not compressed	nil	0.033	not compressed	nil
14	Image14	png	4.37 KB	0.225	not compressed	nil	0.029	not compressed	nil
15	Image15	jpg	31.9 KB	2.165	24.3 KB	0.761	1.791	24.2 KB	0.758
16	Image16	jpeg	8.88 KB	0.894	8.84 KB	0.995	0.804	8.82 KB	0.993
17	Image17	jpeg	8.88 KB	0.806	7.85 KB	0.884	0.83	7.81 KB	0.879
18	Image18	jpeg	10.3 KB	0.772	7.21 KB	0.7	0.74	7.18 KB	0.697
19	Image19	jpg	28.6 KB	0.877	12.4 KB	0.433	0.87	12.4 KB	0.433
20	Image20	jpg	13.8 KB	0.772	10.4 KB	0.753	0.767	10.4 KB	0.753

(Table 2)

We had run the vectorised code in a five, three, two core processors to see the difference in speed (Table 3), thus as the number of processor increases the speed also increases, and thus the purpose of the paper is shown.

Table 3: Comparison of vectorised code in different multi core processors

S.No	Image Name	Image Format	Original Size	vec algo in 2 core processor	vecalgo in 3 core processor	vecalgo in 5 core processor
				Speed (in seconds)	Speed (in seconds)	Speed (in seconds)
1	Image9	jpg	52.7 KB	1.691	1.298	1.28
2	Image10	jpeg	159 KB	3.145	3.141	3.06
3	Image12	jpeg	12.8 KB	0.742	0.544	0.238
4	Image19	jpg	28.6 KB	0.87	0.641	0.293
5	Image20	jpg	13.8 KB	0.707	0.506	0.232

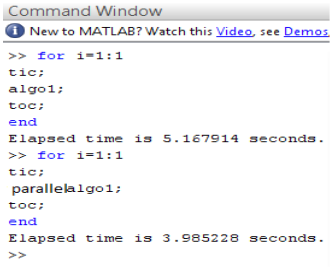
(Table 3)

B .Figures and Tables

The result obtained through our experiment is shown below, (figure 2), (figure 3).We used MATLAB environment for scalar execution time calculation and parallelization.

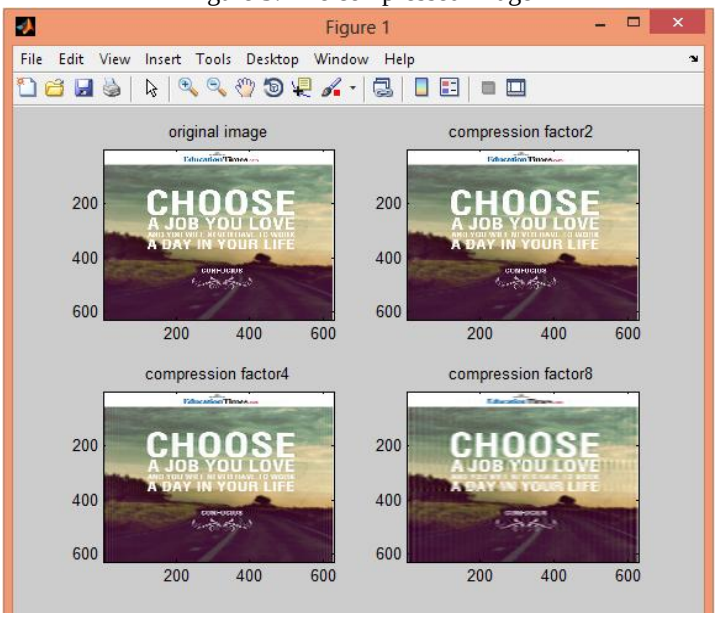
The graph of results obtained is given below (Graph 1), (Graph 2), (Graph 3).

Figure 2: Execution time difference of normal and vectorised code



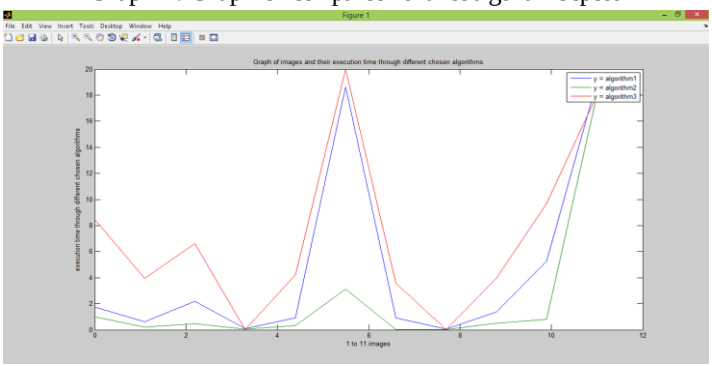
(Figure 2)

Figure 3: The compressed image



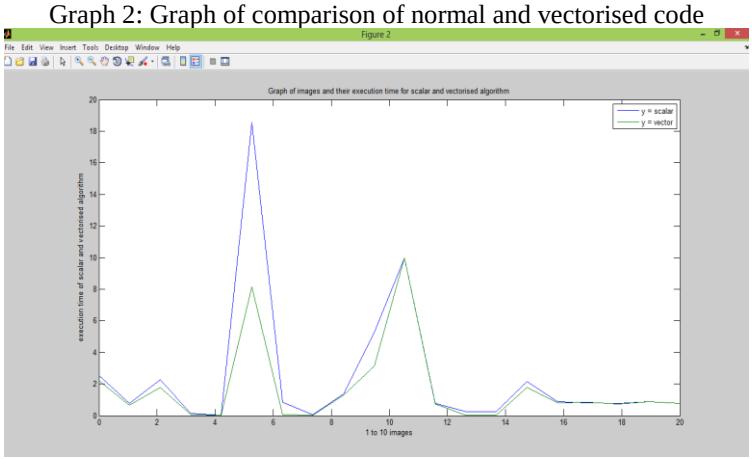
(Figure 3)

Graph 1: Graph of comparison of three algorithms speed



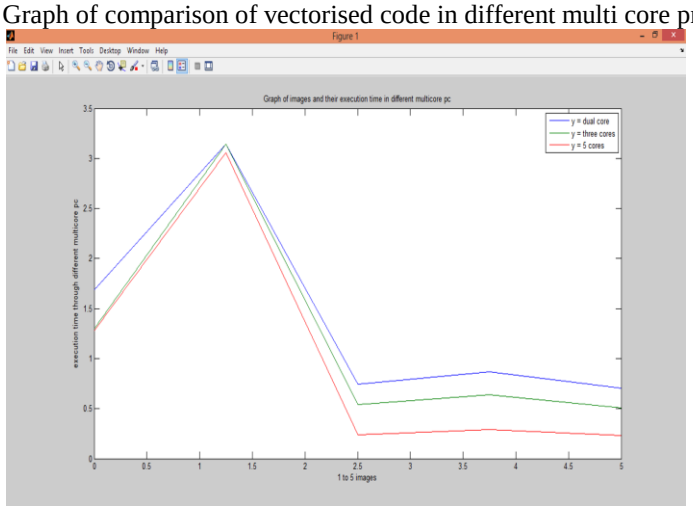
(Graph 1)

Graph 2: Graph of comparison of normal and vectorised code



(Graph 2)

Graph 3: Graph of comparison of vectorised code in different multi core processors



(Graph 3)

III. ADDITIONAL INFORMATION

A. Abbreviations and Acronyms

MATLAB stands for MATrix LABoratory.
DCT stands for Discrete Cosine Transform.

B. Basic outline of the software used (MATLAB)

MATLAB has built in functions for almost all our needs, thus MATLAB code is not long like other programming languages like C, C++.But MATLAB is a little slower when compared to others. It is very suitable to work on Images, hence we chose MATLAB for our project.

IV. CONCLUSION

Thus we have taken an already existing code of a desktop application and tried to optimize the code so that it makes full use of the hardware multi core environment. The same must be taken into consideration by the software developers in future and write codes in such a way that it uses the parallel environment efficiently. In this paper we have made our small step towards it.

ACKNOWLEDGMENT

We would like to thank Prof.Manikandan.N Assistant Prof (Sr), VIT University, Vellore for guiding us through the work, on this paper.

REFERENCES

[1]. Schalkwijk, Jerôme, et al. "High-performance simulations of turbulent clouds on a desktop PC: Exploiting the GPU." *Bulletin of the American Meteorological Society* 93.3 (2012): 307-314.

[2]. Kim, Cheong Ghil, Jeom Goo Kim, and Do Hyeon Lee. "Optimizing image processing on multi-core CPUs with Intel parallel programming technologies." *Multimedia Tools and Applications* (2014): 1-15.

[3]. Ferdman, Michael; Adileh, Almutaz; Kocberber, Onur; Volos, Stavros; Alisafaee, Mohammad; Jevdjic, Djordje; Kaynak, Cansu; Popescu, Adrian Daniel; Ailamaki, Anastasia; Falsafi, Babak, "A Case for Specialized Processors for Scale-Out Workloads," *Micro, IEEE* , vol.34, no.3, pp.31,42, May-June 2014 doi: 10.1109/MM.2014.41

[4]. Method of Optimizing The compression of image data, with automatic selection of compression conditions.

[5]. Evolution of Thread-Level Parallelism in Desktop Applications
Authors:Geoffrey BlakeUniversity of Michigan, Ann Arbor, MI, USARonald G. DreslinskiUniversity of Michigan, Ann Arbor, MI, USA

[6]. The profiling method in multicore processor for effective performance improvement
Authors:Seung Hyun YoonSungkyunkwan University, Suwon, KoreaKyung Min LeeSungkyunkwan University, Suwon, KoreaYong Seok KimSamsung Electronics, Suwon, KoreaSeong Jin ChoSungkyunkwan University, Suwon, KoreaDong Won ChoiSungkyunkwan University, Suwon, KoreaKey Ho KwonSungkyunkwan University, Suwon, KoreaKil Jae KimSungkyunkwan University, Suwon, KoreaJong Hyun ParkSungkyunkwan University, Suwon, KoreaJae Wook JeonSungkyunkwan University, Suwon, Korea

[7]. Kim, Taehwan, et al. "Sequence Data Indexing Method Exploiting the Parallel Processing Resources of GPGPU." (2013).

[8]. Ferdman, Michael; Adileh, Almutaz; Kocberber, Onur; Volos, Stavros; Alisafaee, Mohammad; Jevdjic, Djordje; Kaynak, Cansu; Popescu, Adrian Daniel; Ailamaki, Anastasia; Falsafi, Babak, "A Case for Specialized Processors for Scale-Out Workloads," *Micro, IEEE* , vol.34, no.3, pp.31,42, May-June 2014doi: 10.1109/MM.2014.41