Analytical Evaluation of the 2D-DCT using paralleling processing

Angela Di Serio

Universidad Simón Bolívar
Departamento de Computación y
Tecnología de la Información
Apartado 89000. Caracas, Venezuela

One of the current research areas in the field of computer science is distributed computing systems. In distributed systems, software is partitioned into modules and executed using a number of processors concurrently. A major difficulty in using distributed and paralleling computing systems has been ease of use. There is not a clear methodology for programmers for using these systems effectively. This work seeks to assess the viability of using analytic performance analysis to assist in the evaluation of candidate algorithms through its application to a case study. This will help us to estimate the total execution time and the optimal number of processors.

I. Introduction

A major difficulty in using distributed and parallel computing systems has been ease of use. There is not a clear and simple methodology for programmers for using these systems effectively. This work seeks to assess the viability of using analytic performance analysis to assist in the evaluation of candidate parallel/distributed algorithms through its applications to a case study application to the Discrete Cosine Transform (DCT). Since DCT is computation intensive, we want to focus our attention on developing a DCT algorithm that can be executed on a shared memory multiprocessor architecture considering parallel processing to find an optimal execution and the optimal number of processors.

The Discrete Cosine Transform (DCT) was first proposed by Ahmed *et al.* (1974), and it has been more and more important in recent years. DCT has been widely used in signal processing of image data, especially in coding for compression, for its near-optimal performance. Because of the wide-spread use of DCT's, research into fast algorithms for their implementation has been rather active [2]-[6], and also, since the DCT is computation intensive, the development of high-speed hardware and real-time DCT processor design have been object of research [7]-[9].

This work intends to show different approaches in implementing a two-dimensional DCT using a Sequent system, and to formulate an analytical model for each of these implementations that will help us to estimate the total execution time and the optimal number of processors.

The rest of this paper is organized as follows. Section II gives some background information. Section III proposes an implementation for the one-dimensional DCT that will be used to estimate the two-dimensional DCT, and two different approaches for the two-dimensional DCT. Results are discussed in Section IV. Finally, conclusions are given in Section V.

II. Background

Different approaches have been used trying to find an efficient algorithm for the computation of the one-dimensional DCT and the two-dimensional DCT. Although the approaches and the resulting algorithms are quite different, the main purpose of achieving speed and accuracy is the common goal. In our case, we also want to achieve these goals but using parallel processing. The main purpose of these technologies is to perform computations faster that can be done with a single processor by using a number of processors concurrently.

For a given data sequence $\{x(i): i = 0,1,\ldots,N-1\}$, the one-dimensional DCT sequence $\{Y(j): j = 0,1,\ldots,N-1\}$ is given by

$$Y(m) = \frac{2}{N} k(m) \sum_{n=0}^{N-1} x(n) \cos\left[\frac{(2n+1)m\pi}{2N}\right]$$

where m = 0, 1, ..., N - 1, and

$$k(m) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for m} = 0 \\ 1 & \text{otherwise} \end{cases}$$

The input sequence $\{x(i): i = 0,1,\ldots,N-1\}$ can be represented by the column vector x, and the one-dimensional DCT sequence $\{Y(j): j = 0,1,\ldots,N-1\}$ is given by $Y = (2/N)[A_N]x$ where $[A_N]$ is equal to $k(m)\cos\left[\frac{(2n+1)m\pi)}{2N}\right]$.

Let an (M x N) matrix [g] represent a black-and-white digital picture, where the matrix element g_{mn} may be interpreted as the gray level or intensity of the pixel at the (m,n) location. Let [G] be the two-dimensional DCT of [g]. Then the uv-element of [G] is given by

$$G_{uv} = \frac{2k(u)k(v)}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} g_{mn} \cos\left[\frac{(2m+1)u\pi}{2M}\right] \cos\left[\frac{(2n+1)v\pi}{2N}\right],$$

where u=0,1,...,M-1, v=0,1,...,N-1, and

$$k(i) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } i = 0, \\ 1 & \text{otherwise} \end{cases}$$

A two-dimensional (M x N) DCT can be implemented by M N-point DCTs along the rows of [g], followed by N M-point DCTs along the columns of the matrix obtained after the row transformation.

$$\begin{vmatrix} g_{00} & g_{01} & \cdots & g_{0N-1} \\ g_{10} & g_{11} & \cdots & g_{1N-1} \\ \vdots & \vdots & \cdots & \vdots \\ g_{M-10} & g_{M-11} & \cdots & g_{M-1N-1} \end{vmatrix} \xrightarrow{\text{M N - point DCT}} \begin{vmatrix} g'_{00} & g'_{01} & \cdots & g'_{0N-1} \\ g'_{10} & g'_{11} & \cdots & g'_{1N-1} \\ \vdots & \vdots & \cdots & \vdots \\ g'_{M-10} & g'_{M-11} & \cdots & g'_{M-1N-1} \end{vmatrix}$$

$$\Downarrow \text{ N M - point DCT}$$

$$\begin{vmatrix} G_{00} & G_{01} & \cdots & G_{0N-1} \\ G_{10} & G_{11} & \cdots & G_{1N-1} \\ \vdots & \vdots & \cdots & \vdots \\ G_{M-10} & G_{M-11} & \cdots & G_{M-1N-1} \end{vmatrix}$$

where

$$g'_{uv} = \sqrt{\frac{2}{N}} k(v) \sum_{n=0}^{N-1} g_{un} \cos\left[\frac{(2n+1)v\pi}{2N}\right]$$

$$G_{uv} = \sqrt{\frac{2}{M}} k(u) \sum_{m=0}^{M-1} g'_{mv} \cos\left[\frac{(2m+1)u\pi}{2M}\right]$$

Since loops provide the greatest potential of parallelism to be exploited by multiprocessor systems, it is reasonable and effective to focus our attention on how to determine the optimum degree of parallelism, *i.e.*, how many processors to use to compute the two-dimensional DCT. However, maximum parallel execution, *i.e.*, executing all the processes in parallel may not provide the least time cost solution due to increased communication costs.

III. Implementation using Parallel Processing

The one-dimensional DCT can be easily implemented with two nested loops:

```
for (i=0; i < N; i++)
{
    for (j=0;  j < N; j++)
        Y[i] = Y[i] + x[j] *  cos((2j+1)*iπ/2N)
    Y[i] = 2 * Y[i] / N
 }
```

Each iteration of the outer loop can be considered as a module (DCT(i)), and it can be assigned to a processor. Therefore, in the computation of the DCT for a sequence of size N there are N modules or processes each with execution t_{ID}. A DOALL approach can be used to execute all of the N independent modules in parallel using N processors. A main obstacle would be the un-availability of processors. For this alternative, we want to find the total time cost and the optimum number of processors using some of the results of Garg[10].

If p processors are used to compute the DCT then each processor will execute a maximum of $\lceil N/p \rceil$ modules. F_{cost} is the time cost to execute the fork operation and J_{cost} is the time cost to execute the join operation. On a shared memory multiprocessor like the Sequent Symmetry system, it was found [11] that the cost to create multiple processes was additive and was directly proportional to the number of processors. Therefore, the time cost to create p processes is $p*t_x$, where t_x is the process creation time. t_c is the communication time between modules in a shared memory multiprocessor. In a sequential communication model, the total time to distribute the data is given by $p*N*t_c$. This is because each processor needs the complete data sequence to perform its own computation, and it has to be sent sequentially to each processor. The total time to collect the data is $N*t_c$. The completion time for the parallel DCT in a sequential communication model is given by:

$$T_p = F_{cost} + J_{cost} + \left\lceil \frac{N}{p} \right\rceil * t_{1D} + p * t_x + (p+1) * N * t_c$$

The optimum number of processors to obtain the least-time-cost solution for this implementation is given by:

$$p = \begin{cases} \left\lceil \sqrt{\frac{N * t_{1D}}{t_x + N * t_c}} \right\rceil & \text{if} \left\lceil \sqrt{\frac{N * t_{1D}}{t_x + N * t_c}} \right\rceil < N \\ N & \text{otherwise} \end{cases}$$

If the communication model is a broadcast then the total time to distribute the data is $N*t_c$, since the data set size is N, and the data can be sent at the same time to each of the p processors in $N*t_c$. The total time to collect the data is $N*t_c$. Each process will compute one output value and it has to be collected in a sequential manner. The completion time for the parallel DCT in a broadcast communication model is given by:

$$T_p = F_{cost} + J_{cost} + \left\lceil \frac{N}{p} \right\rceil * t_{1D} + p * t_x + 2 * N * t_c$$

The optimum number of processors to use to execute N modules in order to obtain the least-time-cost solution is given by

$$p = \begin{cases} \left\lceil \sqrt{\frac{N * t_{1D}}{t_x}} \right\rceil & \text{if} \left\lceil \sqrt{\frac{N * t_{1D}}{t_x}} \right\rceil < N \\ N & \text{otherwise} \end{cases}$$

For the two-dimensional DCT, we will show two different approaches. In the first one, each process consists of two steps. The first step computes a N-point one-dimensional DCT over the rows of the original matrix. The second step computes a N-point one-dimensional DCT over the

columns of the matrix obtained after the first step. If p processors are used to compute the two-dimensional DCT then each processor computes a one-dimensional DCT over $\lceil N/p \rceil$ rows, that is, processor 0 computes one-dimensional DCT over rows 0, p, 2p, and so on, processor 1 computes one-dimensional DCT over rows 1, p+1, 2p+1, and so on. When all the processes are completed then each processor computes a one-dimensional DCT over $\lceil N/p \rceil$ columns.

$t_{1D(N)}$ is the execution time of a N-point one-dimensional DCT. F_{cost} is the time cost to execute the fork operation and J_{cost} is the time cost to execute the join operation. The time cost to create p processes is $p * t_x$, where t_x is the process creation time. t_c is the communication time. In a sequential communication model, the total time to distribute the data for the computation of the one-dimensional DCT over the rows is $N^2 * p * t_c$. The total time to collect the data is $N^2 * t_c$. For the computation of the DCT over the columns, some of the data is already in the process, and some of the data is being computed for other processes. Then the total time to move data between the different processes is $N^2 * (p-1) * t_c$.

The completion time for the parallel two-dimensional DCT is given by

$$T_p = F_{cost} + J_{cost} + p * t_x + 2 * \left\lceil \frac{N}{p} \right\rceil t_{1D(N)} + 2 * N^2 * p * t_c$$

The optimum number of processors to use to execute an N x N two-dimensional DCT in order to obtain the least-time-cost solution is given by

$$p = \begin{cases} \left\lceil \sqrt{\dfrac{2 * N * t_{1D(N)}}{t_x + 2 * N^2 * t_c}} \right\rceil & \text{if } \sqrt{\dfrac{2 * N * t_{1D(N)}}{t_x + 2 * N^2 * t_c}} < N \\ N & \text{otherwise} \end{cases}$$

If the communication model is broadcast then the total time to distribute the data is $N^2 * t_c$, since the data size is N^2 and the data can be sent at the same time to each of the p processors. The total time to collect the data is also $N^2 * t_c$. The total time to move data between the different processes after the one-dimensional DCT over the rows is $N^2 * t_c$.

The completion time for the parallel two-dimensional DCT with a broadcast communication model is given by

$$T_p = F_{cost} + J_{cost} + p * t_x + 2 * \left\lceil \frac{N}{p} \right\rceil t_{1D(N)} + 3 * N^2 * t_c$$

In the second approach, we want to change the assignment of work to each processor. Process 0 computes the first element of each one-dimensional DCT (g'_{00}, g'_{10}, g'_{20},...$g'_{N-1\,0}$) over the rows of the original matrix, process 1 computes the second element of each one-dimensional DCT (g'_{01}, g'_{11}, g'_{21},...$g'_{N-1\,1}$) over the rows of the original matrix, and so on. When a process finishes with its work load, it can continue computing the one-dimensional DCT over the results obtained from the previous step. That is, process 0 computes the one-dimensional DCT over (g'_{00}, g'_{10}, g'_{20},...$g'_{N-1\,0}$) obtaining (G_{00}, G_{10}, G_{20},...$G_{N-1\,0}$), process 1 computes the one-dimensional DCT over (g'_{01}, g'_{11}, g'_{21},...$g'_{N-1\,1}$) obtaining (G_{01}, G_{11}, G_{21},...$G_{N-1\,1}$), and so on. In this alternative there is no time delay caused by other processes. Each processor is working independently.

If p processors are used to compute the two-dimensional DCT then each processor computes at least $\lceil N/p \rceil$ one-dimensional DCTs over the rows, and after over the columns.

$t_{1D(N)}$ is the execution time of an N-point one-dimensional DCT. F_{cost} is the time cost to execute the fork operation and J_{cost} is the time cost to execute the join operation. The time cost to create p processes is $p * t_x$, where t_x is the process creation time. t_c is the communication time. In a sequential communication model, the total time to distribute the data for the computation of the

one-dimensional DCT over the rows is $N^2 * p * t_c$. The total time to collect the data is $N^2 * t_c$. There is no need to move data between processes. The completion time for the parallel two-dimensional DCT is given by

$$T_p = F_{cost} + J_{cost} + p * t_x + 2 * \left\lceil \frac{N}{p} \right\rceil t_{1D(N)} + N^2 * (p+1) * t_c$$

The optimum number of processors to use to execute an N x N two-dimensional DCT in order to obtain the least-time-cost solution is given by

$$p = \begin{cases} \left\lceil \sqrt{\frac{2 * N * t_{1D(N)}}{t_x + N^2 * t_c}} \right\rceil & \text{if } \sqrt{\frac{2 * N * t_{1D(N)}}{t_x + N^2 * t_c}} < N \\ N & \text{otherwise} \end{cases}$$

IV. Evaluation and Results

The objective is to show different alternatives for the implementation of the two-dimensional DCT, and the corresponding performance models to express the estimated execution time and the optimum number of processors to achieve the least execution time. In this section, we want to evaluate some of these alternatives and compare them with the corresponding analytical model.

The methodology used to achieve these objectives was:

- Different approaches were implemented on the Sequent system and the actual execution time was recorded.
- The actual execution time data obtained from the implementation of the one-dimensional DCT varying the number of processors and the size of the data was used to estimate the execution time of each individual module, the communication time to distribute and collect data, processor creation time and Fork and Join cost.

- Some of these data were available using TCAS [11], but we used multiple regression to estimate the different values needed.
- The estimated values were used in the two-dimensional DCT models.

Table 1 shows some of the real execution time obtained after varying the number of processors and the data size for the one-dimensional DCT.

Table 1: Execution Time One-dimensional DCT

Processors	N=4	N=5	N=6	N=7	N=8	N=16	N=32	N=64
1	5963	6430	7207	8375	9710	24244	83297	320014
2	20566	20717	21640	21486	22746	29918	60041	178525
3	34276	34500	33953	34635	35156	40099	61967	140741
4	47774	47262	47209	47278	47849	51602	68079	128020
5	60452	60960	61008	61098	61146	64722	78977	125654
6	73753	76334	75590	73991	75579	76185	91894	130978
7	89564	87080	87367	86552	89677	95512	101692	136447

Next, a multiple regression was used to test the accuracy of our analytical model and the following results were obtained:

Iteration execution time *te* 76.72 μ-seconds
Communication Time *tc* 4.74 μ-seconds
Fork and Join and process creation time 13545**number of processors*-8475

with an R^2 of 0.99952 . This R^2 indicates that our model fits the data in more than 99.9 percent of the cases. Figures 1 and 2 show the estimated execution time and the real execution time.

The second approach of the two-dimensional DCT was implemented. In this alternative, there is no inter-processor communication. Each processor works independently. The estimated values obtained for the one-dimensional DCT are used in our model to estimate the execution time and the optimum number of processors.

$$F_{cost} + J_{cost} + p * t_x \approx 13545 * p - 8475$$

$$t_{1D(N)} = N * t_{1D} = N * (N * t_e) = N^2 * 76.72$$
$$t_c \approx 4.74$$

The resulting execution times are showed in Table 2. Figures 3 and 4 show the estimation time and the real execution time for the two-dimensional DCT

Table 2: Execution Time Parallel Processing Two-Dimensional DCT

Processors	N=8	N=16	N=32	N=64
1	87010	646684	5167431	41454933
2	63558	346188	2623922	20818952
3	67637	270813	1824728	14203875
4	70516	212257	1351079	10464345
5	83783	218023	1200738	8520784
6	95826	201847	1053911	7239128
7	110977	213049	904793	6466224

V. Conclusion

The major objective of this work was to use the Discrete Cosine Transform for a case study in effective use of a shared memory multiprocessor using parallel processing. Once we have an analytical model of the execution time that describes the behavior of a specific algorithm, we can estimate the optimum number of processors to be assigned to obtain the least execution time.

Different algorithm structures were presented in this work for the DCT. Some of them were implemented on the Sequent system and compared with their correspondent performance model with good predictive accuracy. The statistical approach proved to be an adequate technique. Given the strong relationship between the total execution time and the number of processors, and the data size, it was possible through the use of multiple linear regression to find the coefficients of the model, such as the communication time, the execution time of each iteration , the process creation time, and the Fork and Join cost to be used in our predictions. Using these primitive

Figure 1. One-dimensional DCT N=16

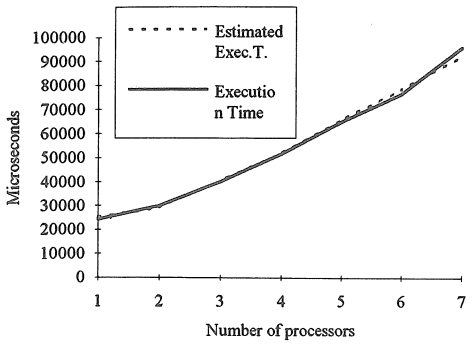


Figure 2. One-dimensional DCT N=32

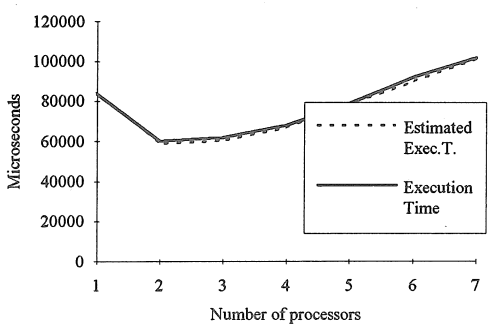


Figure 3. Two-dimensional DCT N=16

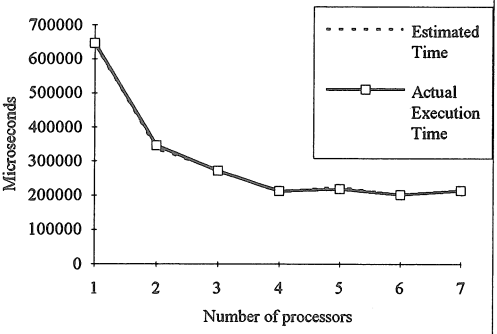
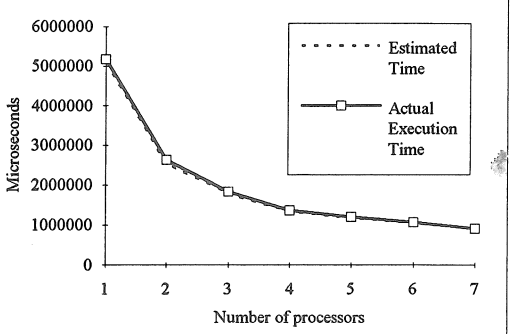


Figure 4. Two-dimensional DCT N=32



function performance estimates, it was possible to explore the behavior of alternative approaches with high accuracy, even beyond the processor limits of the existing Sequent machine.

In the future, it would be interesting to implement the algorithm using pipeling and to compare with parallel processing. It would be also desirable to implement some of the fast algorithms for the DCT using multiprocessors, and compare whether or not the implementation of other single processor algorithms leads to better execution times on multiprocessors systems.

References

[1] N. Ahmed, T. Natarajan, and K.R. Rao, "Discrete Cosine Transform," IEEE Transactions on Computer, vol. C-23, January 1974, pp. 90-94.

[2] W. Chen, C.H. Smith, and S.C. Fralick, "A Fast Computational Algorithm for the Discrete Cosine Transform," IEEE Transactions on Communications, vol. COM-25, N.9, September 1977, pp.1004-1009.

[3] B.G. Lee, "A New Algorithm to Compute the Discrete Cosine Transform," IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-32, N. 6, December 1984, pp. 1243-1245.

[4] Z. Cvetkovic, and M. Popovic, "New Fast Recursive Algorithms for the Computation of Discrete Cosine and Sine Transforms," IEEE Transactions on Signal Processing, Vol. 40, N. 8, August 1992, pp. 2083-2086.

[5] N.I. Cho, and S.U. Lee, "Fast Algorithm and Implementation of 2-D Discrete Cosine Transform," IEEE Transactions on Circuits and Systems, Vol. 38, N. 3, March 1991, pp.297-305.

[6] J. Makhoul, "A Fast Cosine Transform in One and Two Dimensions," IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-28, N. 1, February 1980, pp.27-34.

[7] J. Wu, and W. Duh, "Novel Concurrent Architecture to implement the Discrete Cosine Transform Based on Index Partitions," International Journal of Electronics, 1990, Vol. 68, N.2, pp. 165-174.

[8] C. Chakrabarti, and J. JaJa, "Systolic Architectures for the Computation of the Discrete Hartley and the Discrete Cosine Transforms Based on Prime Factor Decomposition," IEEE Transactions on Computers, Vol. 39, N. 11, November 1990, pp. 1359-1368.

[9] J. Guo, C. Liu, and C. Jen, "A New Array Architecture for Prime-Length Discrete Cosine Transform," IEEE Transactions on Signal Processing, Vol. 41, N. 1 January 1993, pp. 436-442.

[10] S. Garg, "Joint Utilization of Control and Data Partitioning in Determining Performance Optimum Software," PhD's Dissertation, Department of Computer Science and Engineering, University of Connecticut, 1993.

[11] R.G. Hackenberg, "Performance Measurements and Modeling in a Shared Memory Architecture," Master's thesis, Department of Computer Science and Engineering, University of Connecticut, 1992.