Modeling Singular Valued Decomposition (SVD) Techniques using Parallel Programming with pMATLAB.

Miguel Goenaga, Graduate Student, Member, IEEE, Carlos J. González, Graduate Student, Member, IEEE,
Inerys Otero, Graduate Student, Member, IEEE, Juan Valera, Graduate Student, IEEE Member
Domingo Rodríguez, Advisor - Professor, Member, IEEE.

Abstract

In this work certain alternative techniques or methods for computing the singular value decomposition (SVD) of a matrix are pursued in order to study feasibility issues pertaining to multicore embedded computing implementation efforts. The objective of the article is to present a modeling formulation for SVD parallel implementation using pMATLAB. Large scale fast Fourier transform (FFT) operations diagonalize circulant matrices of the order 130^2 by 130^2, through Kronecker products formulations. The modeling and simulation works were performed on a 64-bit, Intel Xeon workstation with 12 Gigabytes of RAM.

Keywords: **parallel Matlab pMatlab Toolbox, Parallel Computing, Kronecker Products, FFT, Singular Value Decomposition (SVD).**

Introduction

This work presents a comparative analysis based in the implementation of the singular value decomposition (SVD) algorithm, using serial and parallel programming. The SVD transform is a powerful technique for matrix factorization where the singular values (which are always positive and real) are the squares of the eigenvalues of the product resulting from the matrix being decomposed times its conjugate transpose. SVD are useful in obtaining minimum norm solutions for linear systems of equations representing filtering applications in areas such as hyperspectral image processing.

Problem Formulation

The SVD of *N by M* matrix, say A, is as follows:

$$A = U_A D_A V_A^T \ ,$$

where $\in C^{N\times M}$, $U_A \in C^{N\times N}$, $D_A \in R^{N\times M}$, $V_A^H \in C^{M\times M}$.

To obtain the SVD of an *N*-by-*M* matrix A, [1] proposed:

$$A = (F_M \otimes I_N)^{-1} U_D D_A V_D^T (F_M \otimes I_P)$$

where *A* is a block circular matrix (CB) [3], *U* is an *N*-by-*M* matrix, *V* is an *M*-by-*M* unitary matrix and D_A is an *M*-by-*M* diagonal matrix with nonnegative entries (setting: *N =M=P),*

$$D = diag(\phi_{1,\dots,}\phi_N),$$

Where *D* is a diagonal matrix carrying the singular values of the CB matrix. According to reference [6], in general, a circulant matrix is related to Fourier diagonal matrix by

$$D_A = F_{NN} A F_{NN}^{-1}$$

Where D_A is the spectrum of the first column of the CB matrix A. Applying a Kronecker property from [4] [5],

$$F_{NN} = (F_N \otimes F_N)$$

yields,

$$D_A = (F_N \otimes F_N) A (F_N \otimes F_N)^{-1}$$

$$A = (F_N \otimes F_N)^{-1} D_A (F_N \otimes F_N)$$

The following Kronecker property

$$AC \otimes BD \Rightarrow (A \otimes B)(C \otimes D) \Rightarrow C = I_m = B;\ A = F_N = D$$

$$(I_N F_N) \otimes (F_N I_N) = (I_N \otimes F_N)(F_N \otimes I_N)$$

links the previous mentioned identity to SVD formulated by [1] in the following manner:

$$A = [(I_N \otimes F_N)(F_N \otimes I_N)]^{-1} D_A [(I_N \otimes F_N)(F_N \otimes I_N)] \ ,$$

yielding

$$A = (F_N \otimes I_N)^{-1} U_D D_A V_D^T (F_N \otimes I_N).$$

Where $U_D = (I_N \otimes F_N)^{-1}$, and $V_D^T = (I_N \otimes F_N)$. Isolating for $U_D D_A V_D^T$ gives

$$U_D D_A V_D^T = (I_N \otimes F_N)^{-1} A (I_N \otimes F_N)$$

Methodology

Using the commutation property from [6], expressed as

$$P_{N,R}(A_S \otimes B_R) P_{N,S} = B_R \otimes A_S,$$

the following transformation was obtained:

$$A = P_{N^2,N}(I_N \otimes F_N)^{-1} P_{N^2,N} U_D D V_D^T P_{N^2,N} (F_M \otimes I_P) P_{N^2,N}.$$

Noticing that $P_{N^2,N} U_D D V_D^T P_{N^2,N} = U_D D V_D^T$, results in our target SVD formulation:

$$A = P_{N^2,N}(I_N \otimes F_N)^{-1} U_D D_A V_D^T (I_N \otimes F_N) P_{N^2,N}$$

The implementation of the parallel algorithm was based on the execution time analysis of its serial implementation by making emphasis on its critical points (most delayed execution times). The calculation of the circulant block matrix with its respective circulant blocks and the multiplication of matrices through Kronecker products formulations were parallelized. In particular, it is shown how the local instances can be used to accelerate the slowest part of the algorithm – updating the block-columns, and the local instances multiplying matrices with block-columns, respectively.

Results

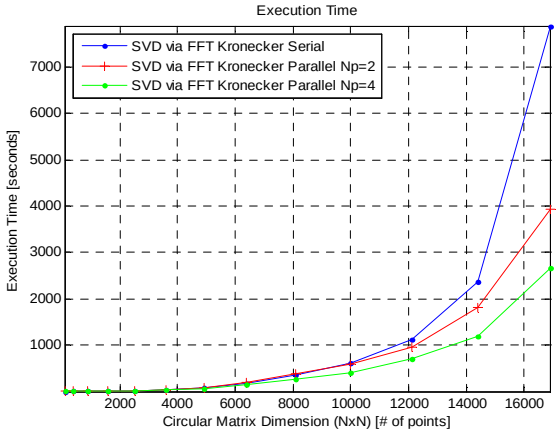
Two dimensional arrays $h[n_0, n_1] \in (Z_N x Z_N)$ with arbitrary normal distribution $\varkappa(0,1)$ were generated to produce circular matrices $A \in (Z_{N^2} x Z_{N^2})$ for different sizes of *N*.

Execution times were documented and displayed graphically for different array sizes via serial and parallel processes.

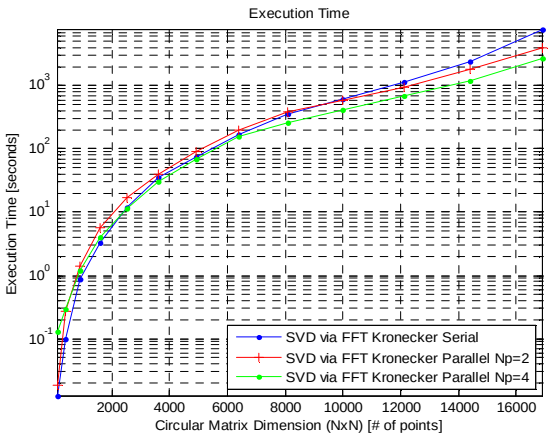
Table 1 presents the execution times obtained for the serial and parallel implementations. ***Figure 1*** presents execution times using pMATLAB, with two and four processors.

Table 1: Serial & Parallel SVD Execution Time Results

Circular Matrix A # of points of ($N^2 \times N^2$)	SVD Execution times using Kronecker Products. (serial) (s)	SVD Execution times using Kronecker Products. (pMATLAB) (s), Np = 2	SVD Execution times using Kronecker Products. (pMATLAB) (s) Np = 4
100 × 100	0.012678	0.0186	0.1276
400 × 400	0.097356	0.2693	0.2963
900 × 900	0.889663	1.4016	1.2135
1600 × 1600	3.285304	5.6016	4.0699
2500 × 2500	11.721894	16.7797	11.3476
3600 × 3600	35.008011	40.4735	30.6647
4900 × 4900	74.42742	90.6983	68.317
6400 × 6400	168.083828	195.138	157.2951
8100 × 8100	348.144266	389.1622	263.8286
10000×10000	621.572049	589.4422	413.5672
12100×12100	1129.08065	956.362	695.6591
14400×14400	2367.406236	1818.6529	1185.2365
16900×16900	7880.914234	3937.6814	2652.5622



a) Normal



b) Logarithmic

Figure 1: Execution time comparison between SVD computations using Pmatlab.

Conclusions

Through the increasing influence of parallel High Performance Computing (HPC), it has become feasible to create large systems with an ever-increasing number of variables. Present investigation efforts seek to learn more about iterative solvers in order to efficiently address even larger systems. The contribution of this paper is the proposal and implementation of a robust SVD solver using the Kronecker products methodology in a parallel architecture environment, with pMATLAB being used as a modeling and simulation tool-aid for parallel programming.

References

[1] Cao-Huu, T. and Evequoz, C. "*Singular value decomposition transform with and FFT-based algorithm on the Connection Machine CM5,*" Electrical and Computer Engineering, 1995. Canadian Conference, Volume: 2, page(s): 1046-1049 vol.2, 5-8 Sep 1995.

[2] Chen, L. and Yap, K. H. "*Regularized Interpolation Using Kronecker Product for Still Images,*" Image Processing, 2005. ICIP 2005. IEEE International Conference, Volume: 2, page(s): II- 1014-17, 11-14 Sept. 2005.

[3] Rjasanow, S. "*Effective Algorithms With Circulant-Block Matrices,*" Linear algebra and its applications, Volume. 202, Pages. 55-69, 1994.

[4] Van Loan, C. F. "*The ubiquitous Kronecker product,*" Journal of Computational and Applied Mathematics, Volume 123, Issue 1-2, Pages: 85 - 100 November 2000.

[5] Kamm, J. and Nagy J. "*Kronecker product and SVD approximations in the image restoration*", Linear Algebra and its Applications, Volume 284, Issues 1-3, 15 November 1998, Pages 177-192.

[6] Johnson, J. Johnson, R. Rodriguez, D. Tolimieri, R. "*A Methodology for Designing, Modifying, and Implementing Fourier Transform Algorithms on Various Architectures,*" Journal of Circuits, Systems and Signal Processing, Vol. 9, No. 4, pp. 449-500, Birkäuser, 1990.

[7] Boutsidis, C. and Gallopoulos, E. "*SVD based initialization: A head start for nonnegative matrix factorization*", Journal of the Pattern Recognition Society, Science Direct, Pattern Recognition, Vol. 41, pp. 1350-1362, September 20, 2007.

[8] Tsitsas, N. Alivizatos, E. and Kalogeropoulos, G. "*A recursive algorithm for the inversion of matrices with circulant blocks*", Journal of Applied Mathematics and Computation, Science Direct, Vol. 188, pp. 877-894, October, 2007.

[9] Xu, W. and Qiao, S. "*A fast symmetric SVD algorithm for square Hankel Matrices*", Journal of Linear Algebra and its applications, Vol. 428, pp. 550-563, June 03, 2007.