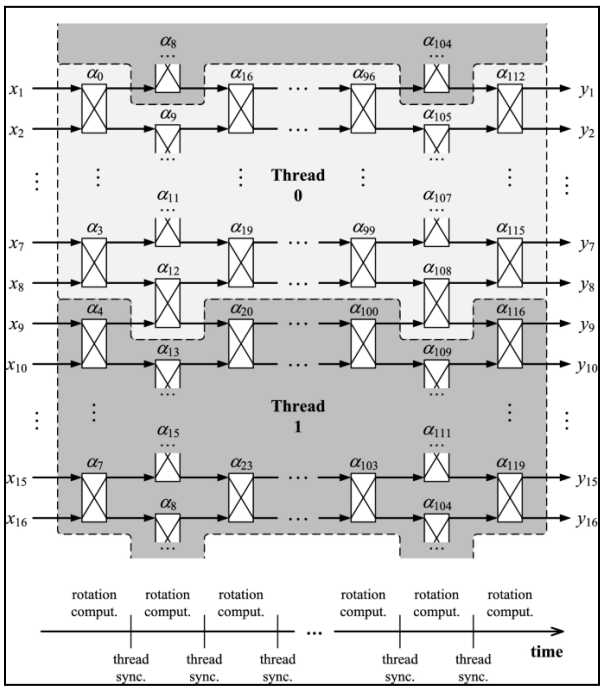
CS205 Team 19
Sibi Raja, Matthew Villaroman, Eliel Dushime, Bowen Zhu

Parallelization of SVD & eigenvalue computation of higher dimension matrices

Our project idea definitely offers data-level parallelism and possibly task-level parallelism. The data-level parallelism comes from performing operations on different portions of the input matrix (i.e. columns) in parallel for SVD. We need to explore more into the parallelization of SVD to understand if and how tasks can be parallelized. The computational bottleneck of performing SVD on a matrix is the sequential orthogonalization process. We want to explore the parallelization of the orthogonalization process. Doing so has the potential to optimize the computational process of several other algorithms like Hestenes, Jacobi, and Kogbetliantz.

We plan to exploit this parallelism by considering distinct columns of a matrix and applying rotation matrices to the columns in parallel for the orthogonalization process. We aim to focus on shared memory models. For example, a given matrix will be placed in shared memory while different threads will each read and update certain columns of the matrix in parallel. We think we will need MIMD because we expect threads to be working on different pairs of data (i.e. different pairs of columns of the original matrix). We are not fully sure if the optimized process of orthogonalization will require multiple instructions for each thread, so we will use MIMD for now.



$$\left(a_i^{(k+1)}, a_j^{(k+1)}\right) = \left(a_i^k, a_j^k\right) \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

The size of the problem we expect to work with is to start with a 10,000 x 10,000 size matrix (we will possibly explore higher dimension matrices later on). In such a matrix, there will be 10^8 floating point numbers, so we estimate 800 MB will be needed. From lab 1, we know that there is 126 GB of memory per node on the compute cluster, so our chosen size of the problem is reasonable.

Performing SVD once will take *O(m*n2*log(n))* on a m x n matrix. No, our solution does not depend on a high volume of I/O since we just consider one matrix.

References:

https://maths-people.anu.edu.au/~brent/pd/rpb128tr.pdf
https://link.springer.com/chapter/10.1007/978-3-030-43229-4_48