Performance Evaluation of the Matlab PCT for Parallel Implementations of Nonnegative Tensor Factorization

Tabitha Samuel, Master's Candidate
Dr. Michael W. Berry, Major Professor

Abstract: Increasingly large datasets acquired by NASA for global climate studies demand larger computation memory and higher CPU speed to mine out useful and revealing information. While boosting the CPU frequency is getting harder, clustering multiple lower performance computers thus becomes increasingly popular. This prompts a trend of parallelizing the existing algorithms and methods by mathematicians and computer scientists. In this PILOT study, I take on the task of parallelizing the Nonnegative Tensor Factorization (NTF) method for climate data, using the Parallel Computing Toolbox in Matlab. The Parallel Computing Toolbox lets you solve computationally and data-intensive problems on multicore and multiprocessor computers. Parallel processing constructs such as parallel for-loops and code blocks, distributed arrays, parallel numerical algorithms, and message-passing functions let you implement task- and data-parallel algorithms in MATLAB at a high level without programming for specific hardware and network architectures. I use methods provided in the Toolbox such as the Parfor loops and Distributed Jobs, for the Nonnegative Tensor Factorization code and evaluate and compare the results obtained from the different approaches over the performance achieved by the serial code.

1. What is the Parallel Computing Toolbox

The Parallel Computing Toolbox lets you solve computationally and data-intensive problems using MATLAB and Simulink on multicore and multiprocessor computers. Parallel processing constructs such as parallel for-loops and code blocks, distributed arrays, parallel numerical algorithms, and message-passing functions let you implement task- and data-parallel algorithms in MATLAB at a high level without programming for specific hardware and network architectures. As a result, converting serial MATLAB applications to parallel MATLAB applications requires few code modifications and no programming in a low-level language. You can run your applications interactively or offline, in batch environments. Some of the key features of the Parallel Computing Toolbox are:

- It solves computationally and data-intensive problems using MATLAB and Simulink on multicore and multiprocessor computers
- It provides support for data-parallel and task-parallel application development
- It provides high-level constructs such as distributed arrays, parallel algorithms, and message-passing functions for processing large data sets on multiple processors
- It can be integrated with MATLAB Distributed Computing Server for cluster-based applications that can use any scheduler or any number of workers

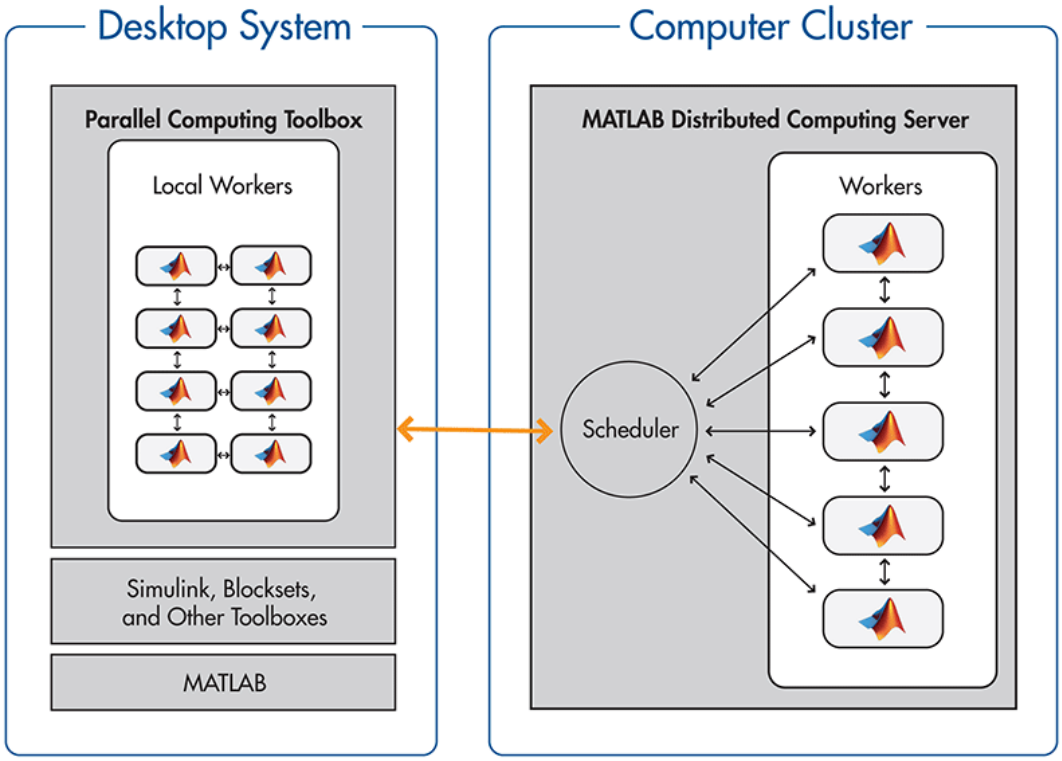

Fig: Setup of Matlab's Parallel Computing Toolbox

2. Application Areas of the Parallel Computing Toolbox

- Parallel for loops

  Parallel Computing Toolbox software improves the performance of loop execution by allowing several MATLAB workers to execute individual loop iterations simultaneously. For example, a loop of 100 iterations could run on a cluster of 20 MATLAB workers, so that simultaneously, the workers each execute only five iterations of the loop. You might not get quite 20 times improvement in speed because of communications overhead and network traffic, but the speedup should be significant.

- Offloading work

  When working interactively in a MATLAB session, you can offload work to a MATLAB worker session to run as a batch job. The command to perform this job is asynchronous, which means that your client MATLAB session is not blocked, and you can continue your own interactive session while the MATLAB worker is busy evaluating your code.

- Large Data sets

  If you have an array that is too large for your computer's memory, it cannot be easily handled in a single MATLAB session. Parallel Computing Toolbox software allows you to distribute that array among multiple MATLAB workers, so that each worker contains only a part of the array. Yet you can operate on the entire array as a single entity. Each worker

operates only on its part of the array, and workers automatically transfer data between themselves when necessary, as, for example, in matrix multiplication.

3. Parallel Implementation of the Nonnegative Tensor Factorization

Data mining techniques are commonly used for the discovery of *interesting patterns.* The code which I'm seeking to find improvement in performance sought to identify regions (or clusters) of the earth which have similar short- or long-term characteristics. Eigensystem-based analysis driven by principal component analysis (PCA) and the singular value decomposition (SVD) has been used to cluster climate indices. But the orthogonal matrix factors generated by the SVD are difficult to interpret. Among other data mining techniques, Nonnegative Matrix Factorization (NMF) has attracted much attention.

In NMF, an $m \times n$ (nonnegative) mixed data matrix *X is approximately factored into a product of* two nonnegative rank-*k matrices, with k small compared to m and n,* $X \approx WH$. *W and H can provide a physically* realizable representation of the mixed data *W and H can provide a physically* realizable representation of the mixed data. Nonnegative Tensor Factorization (NTF) is a natural extension of NMF to higher dimensional data. In NTF, high-dimensional data, such as 3D or 4D global climate data, is factored directly and is approximated by a sum of rank-1 nonnegative tensors.

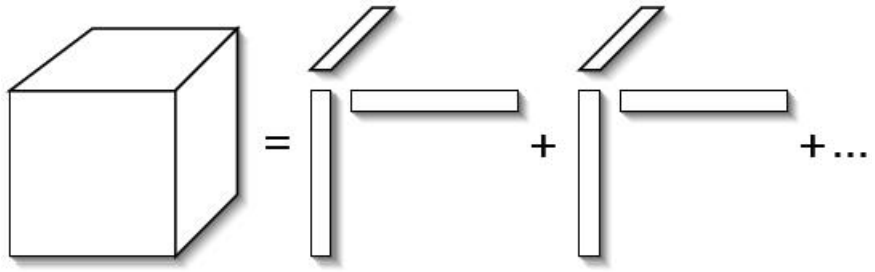

Fig.: An illustration of 3-D tensor approximate factorization using a sum of rank one tensors

We use the Projected Gradient Descent Method to solve the semi-NMF problem. For this we use two quadratic forms of W and A ie. W^TA and W^TW. Comparing the sizes of the two quadratic forms, ie. k*k and k*n with the sizes of W and A ie m*k and m*n and knowing m,n >> k, we can save memory required to store these matrices.

Focus of this PILOT study: To parallelize the computation of W^TA

4. Data involved in the study

Six climate based indices were used for this study.

Name	Description	Adjustment
sst	sea surface temperature	+273.15
ndvi	normalized difference vegetation index	+0.2
tem	land surface temperature	+273.15
pre	Precipitation	
hg500	geopotential height (elevation) for barometric pressure of 500 millibars	+300
hg1000	geopotential height (elevation) for barometric pressure of 1000 millibars	+300

Data from each index was preprocessed. Preprocessing included interpolation to compensate for the sparsity of data and shifting of data to remove negativity of values. Each parameter is represented by a 3-array data cube of size 720 x 360 x 252. The first two dimensions stand for the latitude and longitude, and the third dimension measures the number of months. The timeline used for this study is between January 1982 to December 2002 for a total of 252 months.

5. Interesting patterns derived from this climate data

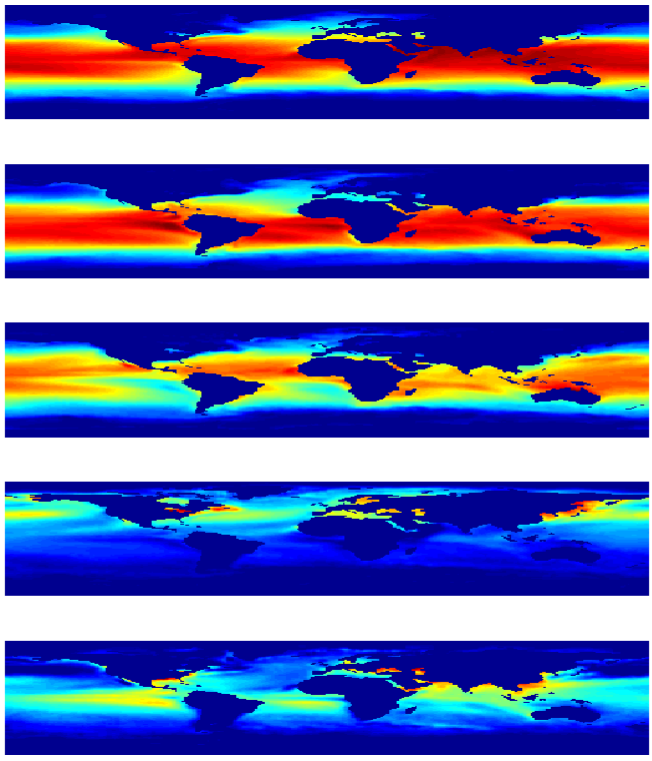

Fig: Global map of sea surface temperature patterns

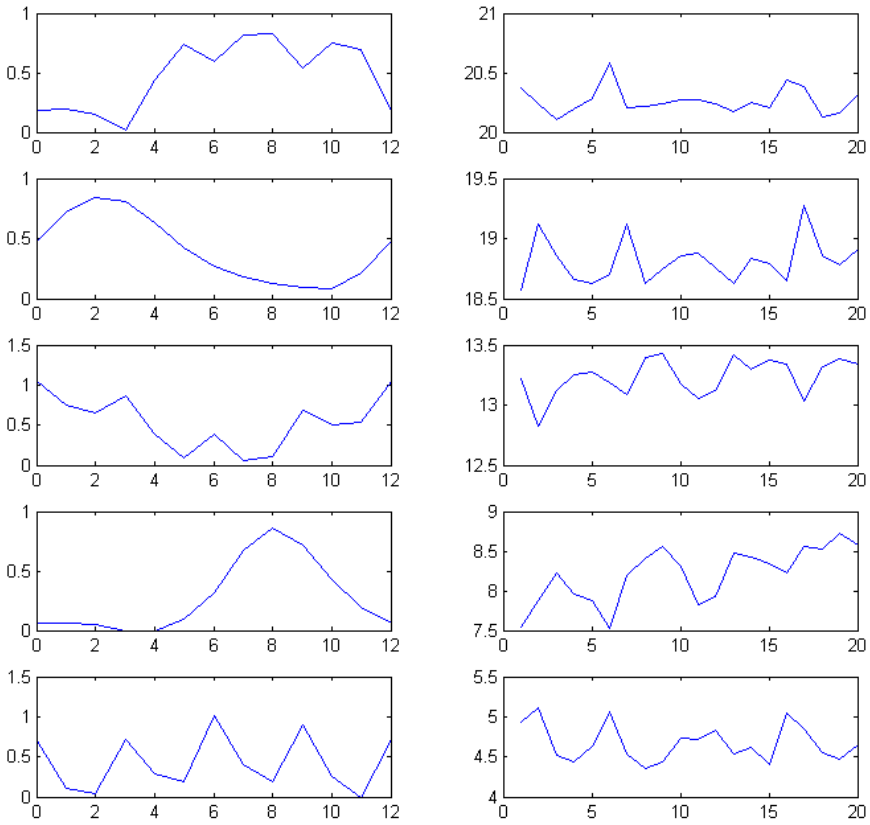

Fig: Monthly and yearly variations of sea surface temperature patterns

6. Approaches Used

a. Parfor Loops
b. Distributed Jobs with slicing A
c. Load and Save with Distributed Jobs

Parfor Loops

- Part of the loop is executed on client, rest on the worker
- Data sent from client to workers, calculations are performed on workers, results are sent back to client where they are pieced together

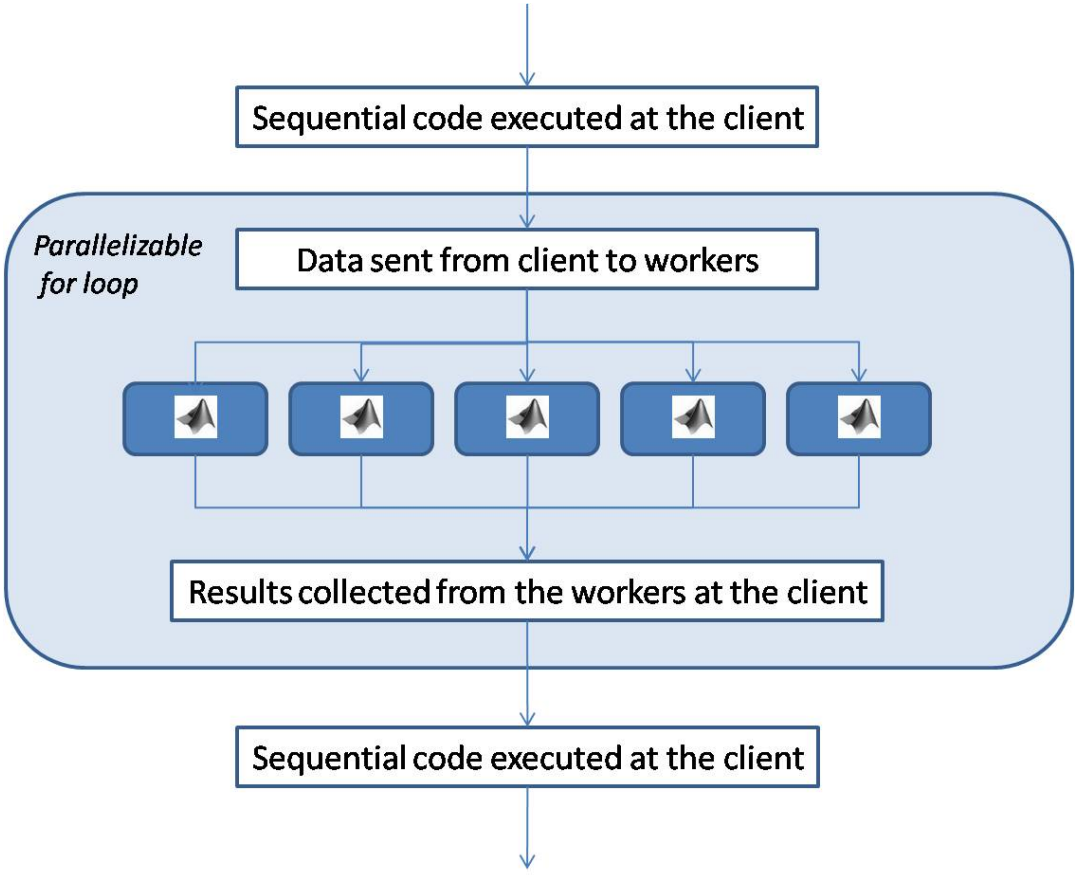

Fig: Code execution using Parfor loop

Distributed Jobs with Slicing A:

In a distributed job:

- Tasks do not directly communicate with each other
- A worker may run several of these tasks in succession
- All tasks perform the same function in a parallel configuration

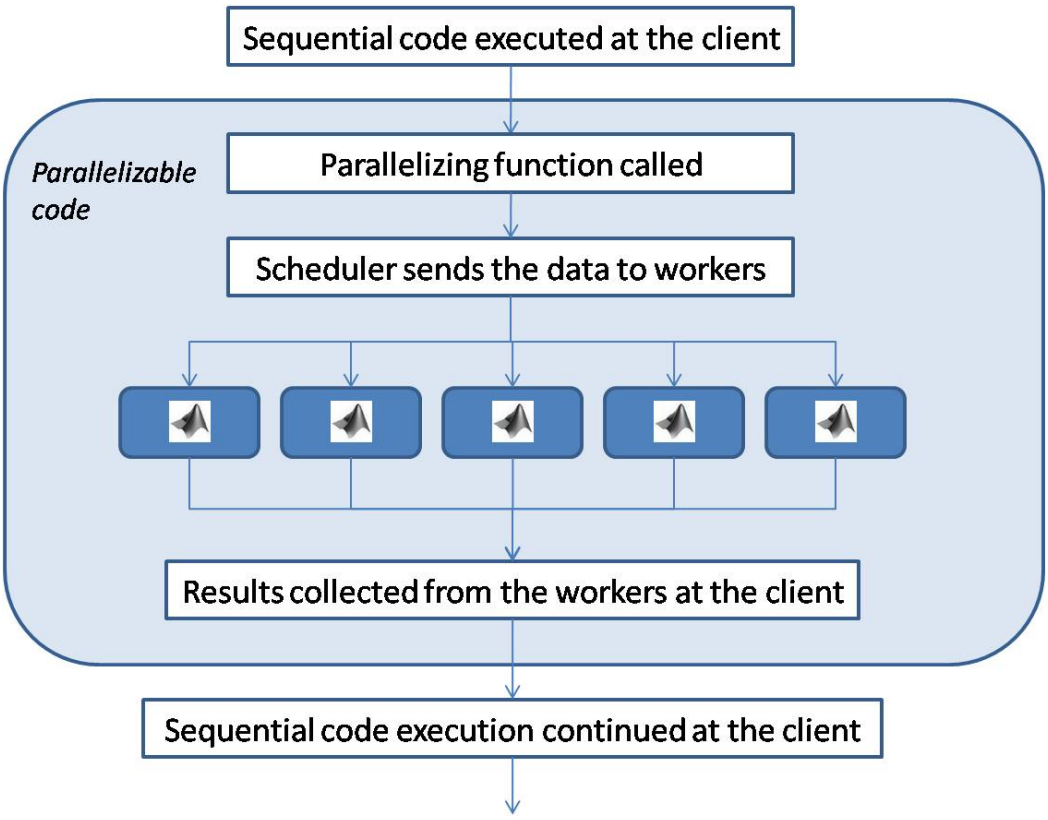

Fig: Code execution using Distributed Jobs

Load and Save with distributed Jobs

- Size of matrix A is very large and linear. For entire dataset the size of A is 65318400 x 1
- In this approach, A is saved to the local workspace of the node, prior to task creation and is reloaded only when there is a change in the value of A

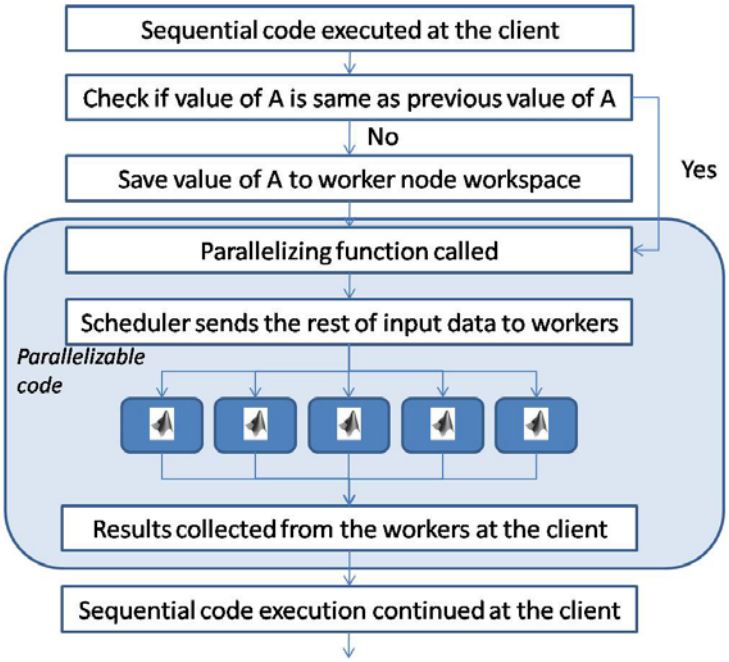

Fig: Code execution at client using Load and Save

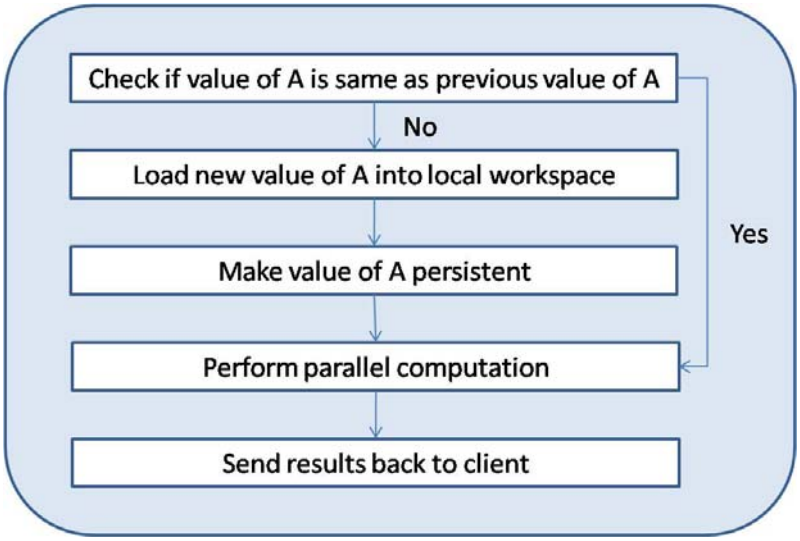

Fig. : Code execution at worker using Load and Save

The PILOT presentation will present the results obtained and conclusions drawn by using these three methods and comparing them to the results obtained by running them on the serial version of the code.

References:

``Parallel Nonnegative Tensor Factorization Algorithm for Mining Global Climate Data,'' Q. Zhang, M.W. Berry, B.T. Lamb, and T. Samuel, Proceedings of the International Conference on Computational Science (ICCS 2009) GeoComputation Workshop, Baton Rouge, LA, Lecture Notes in Computer Science (LNCS) 5545, G. Allen et al. (Eds.), Springer-Verlag, Berlin, (2009), pp. 405-415.

''Scenario Discovery Using Nonnegative Tensor Factorization'', Brett W. Bader, Andrey A. Puretskiy, and Michael W. Berry, in Progress in Pattern Recognition, Image Analysis and Applications, Proceedings of the Thirteenth Iberoamerican Congress on Pattern Recognition, CIARP 2008, Havana, Cuba, Lecture Notes in Computer Science (LNCS) 5197, Jos'e Ruiz-Shulcloper and Walter G. Kropatsch (Eds.), Springer-Verlag, Berlin, (2008), pp. 791-805.

``Discussion Tracking in Enron Email Using PARAFAC'', Brett W.
Bader, Michael W. Berry, and Murray Browne, in Survey of Text Mining II: Clustering, Classification, and Retrieval, M.W. Berry and M. Castellanos (Eds.), Springer-Verlag, London, (2008), pp. 147-163.

``Nonnegative Matrix and Tensor Factorization for Discussion Tracking'', Brett W. Bader, Michael W. Berry, and Amy N. Langville, in Text Mining: Theory, Applications, and Visualization, A. Srivastava and M. Sahami (Eds.), Chapman & Hall/CRC Press, (2010), to appear.