**Distributed Machine Learning**

Table of Contents

1. Abstract……………………………………………………………………………… 1
2. Introduction…………………………….……………………………………………. 2
3. Algorithms and Performance Metrics………………………………………………. 4
4. Conclusion…………………………...……………………………………………..... 8
5. References…………………………………………………………………………… 8
6. **ABSTRACT**

Data generation is increasing at an exponential rate. This has led to an increase in usage of distributed computing as data cannot be stored on a single node and executed in a timely manner. A distributed environment is created using the dask framework on Uconn’s HPC and few machine learning algorithms are executed in a distributed fashion. Performance Metrics are compared by running algorithms using distributed fashion and on a single node.

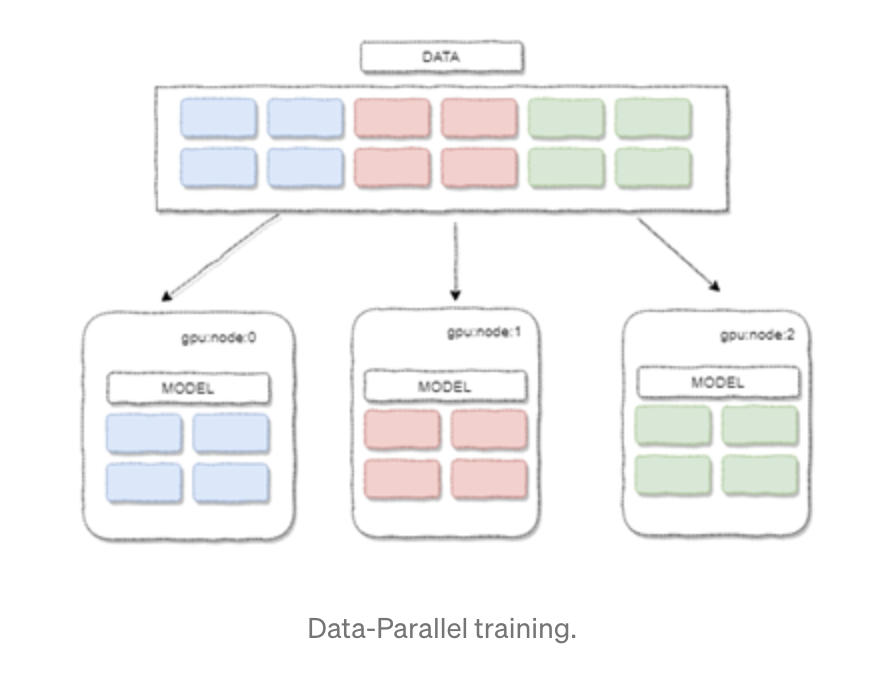
1. **INTRODUCTION**

**2.1 Distributed Machine learning**

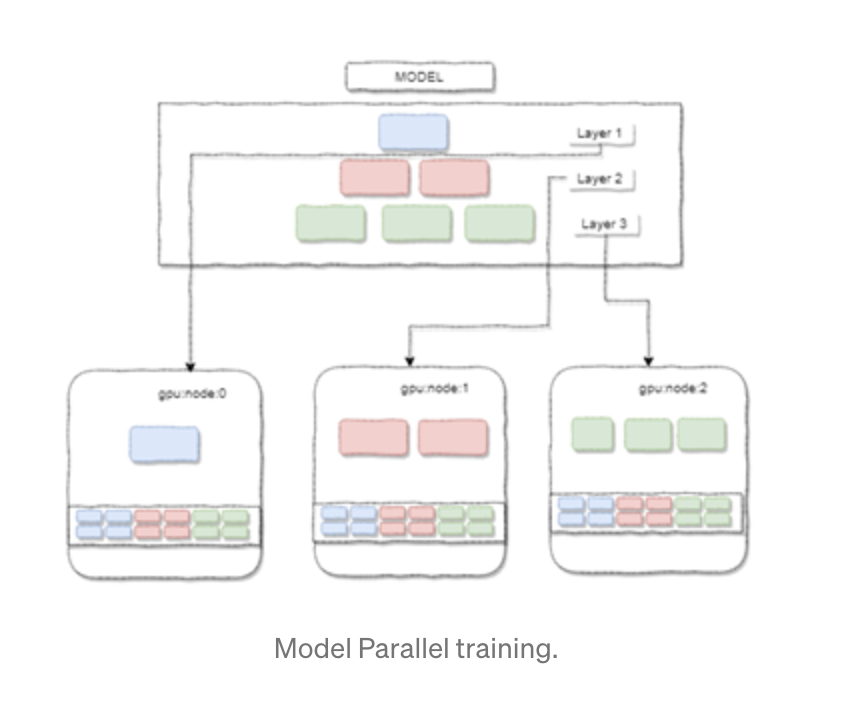
As the data size increases, there will be millions of parameters which will define a model. Usually it takes days to train a model. In some scenarios, data will be too huge to fit on a single machine. Whereas distributed machine learning algorithms handle big data and develop scalable and efficient algorithms with respect to accuracy, execution time and memory usage. Distributed machine learning is termed as a multi node machine learning system that aims to improve the performance and provides scalability. There are two types of distributed ML[2]. They are:

* Data parallelism
* Model parallelism

Data parallelism: When the whole data set doesn't fit on a single machine, then the data is distributed across multiple machines/cores. It reduces the number of I/O operations required to train the model. Therefore, it achieves faster training time.

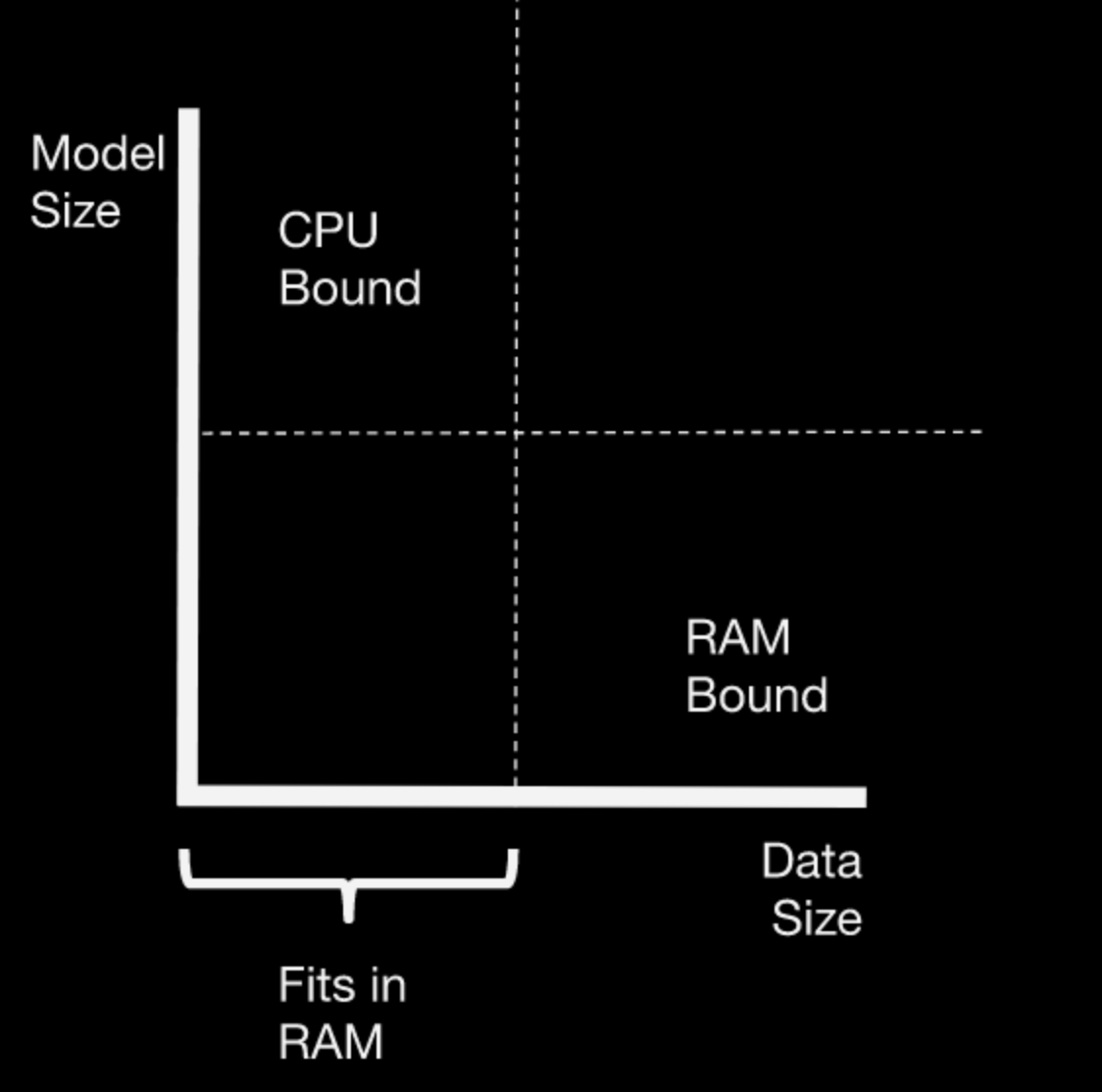


Model parallelism: Model parallelism is performed when the size of the model is too big to fit on a single machine. In model parallelism, the model is split across multiple cores or machines. This process is followed only when the model doesn’t fit in the memory and will not highly speed up the training process.



**2.2 Dask**

Dask is a library for parallel and distributed computing in python. Dask is a very good library to handle large datasets and large models. Dask collections such as dask arrays, dask bag etc., can store the data which is larger than RAM.



Scaling issue will either be due to memory bond(data size) or due to CPU bond(model size). Model size issue is handled by using a dask cluster to parallelize the load on multiple cores. Parallelization is done either through the dask joblib backend or Dask-ML estimators. Data size issue is addressed by high level dask collections (dask arrays, dask bag, dask frame) combined with dask-ml estimators.

Supervised and unsupervised machine learning algorithms mentioned below are executed using dask on Uconn’s HPC

* K-Means
* Linear Regression
* Spectral clustering

1. **ALGORITHMS AND PERFORMANCE METRICS**

**3.1 K-Means:**

K-Means is also known as Lloyd’s algorithm. It is an unsupervised machine learning algorithm K-means will partition n observations into K clusters such that every observation belongs to the cluster having the nearest distance from the centroid. Initialization is done randomly by Forgy method and Random Partition method. In the Forgy method, K observations are randomly chosen and they are used as an initial means. In the Random partition method, clusters are randomly assigned to each observation and later it will proceed to the update step.

Basic K-Means algorithm

| Input: K= no.of clusters, **Points** {x1 ... xn}  **Repeat** **until** **convergence** {  For all points {x1 ... xn}  Find nearest centroid and assign x to cluster C   For each cluster (c1 to Ck)  Recompute the centroid by calculating mean of the points in the cluster  } |
| --- |
|  |

# Dask K-Means algorithm[5] is executed in a parallel and distributed fashion by using Dask-ML and below are the performance metrics of K-Means algorithm combined with Dask which is executed on Uconn’s HPC.

| **Data points** | **No. of cores** | **Execution time** | **Memory usage** |
| --- | --- | --- | --- |
| 10000000 | 1 | 12:15 mins | 3681740K |
| 10000000 | 10 | 3:4 mins | 4888812k |
| 10000000 | 15 | 2:24 mins | 6031340k |
| 10000000 | 36 | 6:48 mins | 10243828k |

**3.2 Linear Regression**

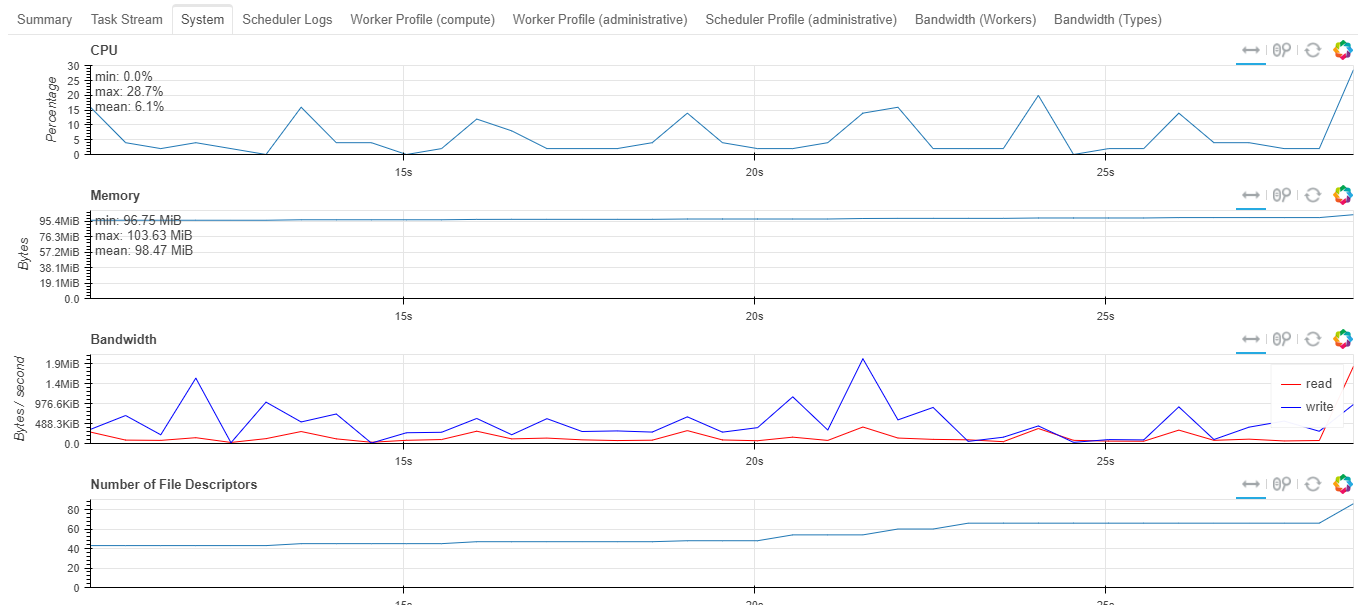
Linear regression is a supervised machine learning algorithm. It attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable.

*Gradient Descent* is the process of minimizing a function by following the gradients of the cost function [3].This involves knowing the form of the cost as well as the derivative so that from a given point you know the gradient and can move in that direction, e.g. downhill towards the minimum value.

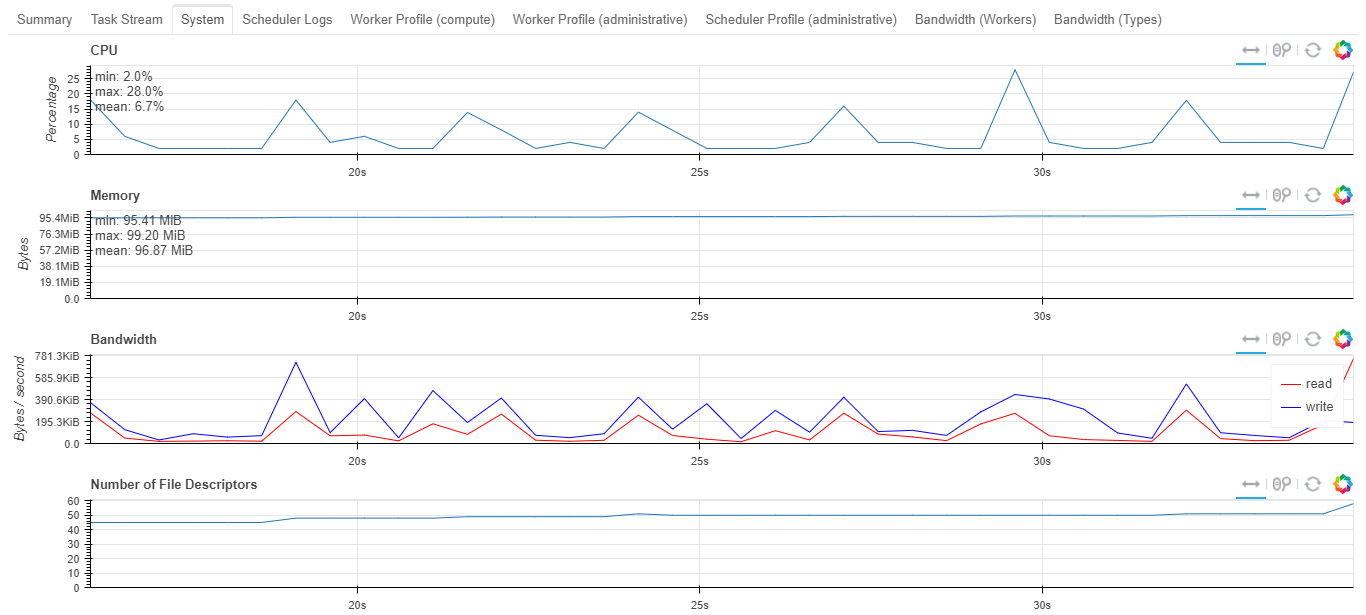
| Data Points | No of Cores | Execution Time | Memory Usage |
| --- | --- | --- | --- |
| 1000000 | 1 | 00:10:41 | 82056K |
| 1000000 | 10 | 00:00:33 | 2268K |
| 1000000 | 15 | 00:03:23 | 8774K |
| 1000000 | 36 | 00:36:25 | 96680K |

**Performance Charts**

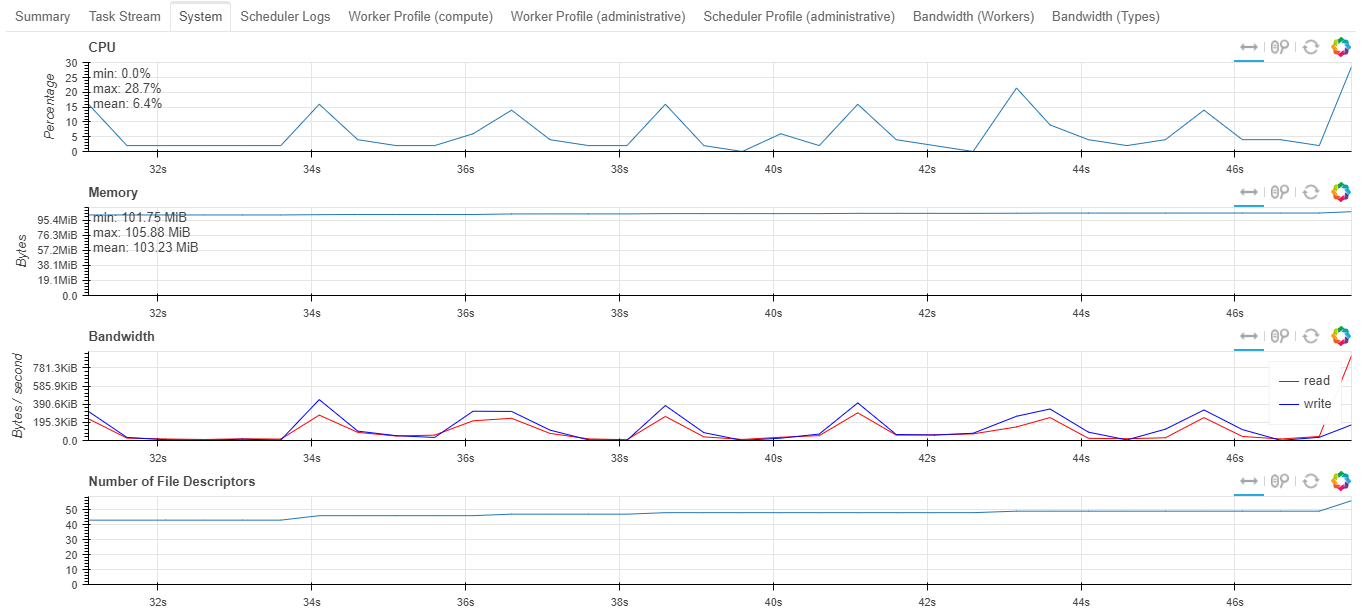
*10 Nodes*

****

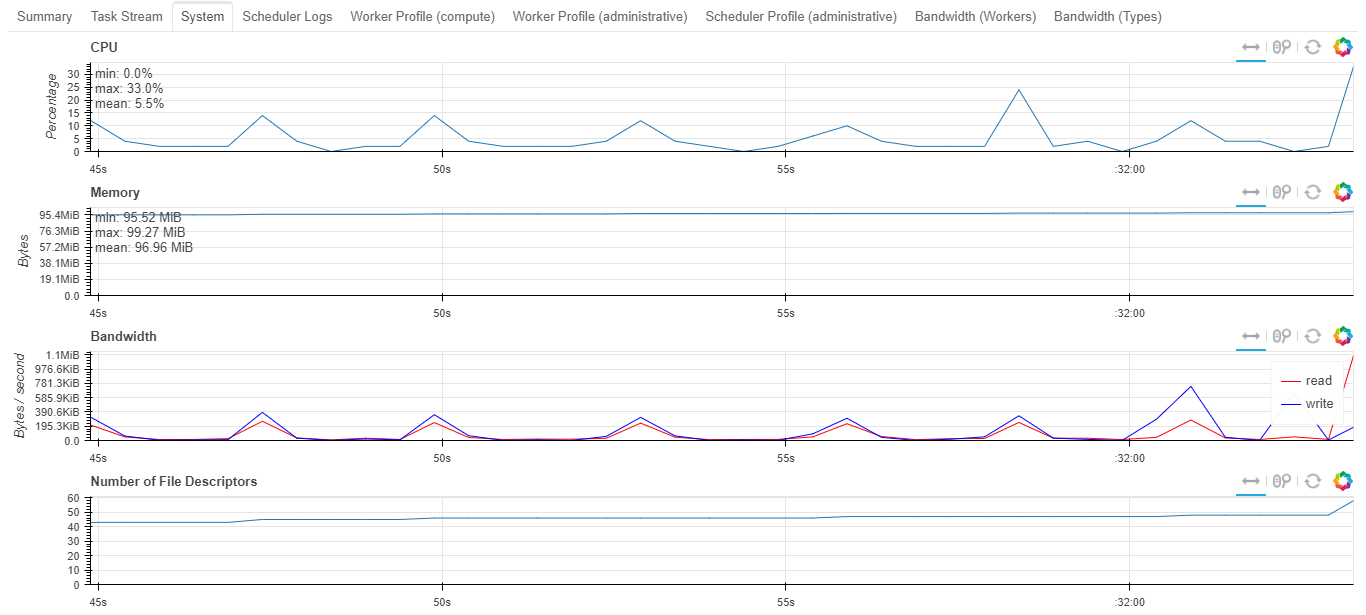
*20 Nodes*

****

*36 Nodes*

****

*1 Node*

****

**3.3 Spectral Clustering**

Spectral clustering clusters based on pairwise proximity/similarity/affinity and have roots in graph theory.Given n data points x\_1, … , x\_n, the spectral clustering algorithm constructs a similarity matrix S, where Sij >= 0 reflects the relationship between xi and xj. It then uses similarity information to group x\_1, ... , x\_n into k clusters[4].

**Algorithm 1**. Spectral clustering using a sparse similarity matrix

Input: Data points x\_1, ... , xn; k: number of desired clusters.

1) Construct similarity matrix S.

2) Modify S to be a sparse matrix.

3) Compute the Laplacian matrix L by (2).

4) Compute the first k eigenvectors of L; and construct V, whose columns are the k eigenvectors.

5) Compute the normalized matrix U of V by (4). 6) Use the k-means algorithm to cluster n rows of U into k groups.

| Data Points | No of Cores | Execution Time | Memory Usage |
| --- | --- | --- | --- |
| 1000000 | 1 | 00:04:44 | 1146412K |
| 1000000 | 10 | 00:01:14 | 1245224K |
| 1000000 | 20 | 00:01:09 | 1690796K |
| 1000000 | 36 | 00:00:36 | 1842684K |

**5. Conclusion**

From the above experimental analysis and results, it can be seen that executing machine algorithms in a distributed environment took less time and executing same algorithms on a single core took more time. However, executing machine algorithms on more cores does not make a huge difference to the runtime as communication cost overhead will be added.

**6. REFERENCES**

**[1].** “**Distributed Machine Learning with Python and Dask**”, [Pablo Salvador Lopez](https://pablo-81685.medium.com/?source=user_profile----------------------------------------)

[**https://towardsdatascience.com/distributed-machine-learning-with-python-and-dask-2d6bae91a726**](https://towardsdatascience.com/distributed-machine-learning-with-python-and-dask-2d6bae91a726)

**[2] “Parallel and Distributed Deep Learning”,**Vishakh Hegde,Sheema Usmani

[**https://web.stanford.edu/~rezab/classes/cme323/S16/projects\_reports/hedge\_usmani.pdf**](https://web.stanford.edu/~rezab/classes/cme323/S16/projects_reports/hedge_usmani.pdf)

[3] “**Lecture notes - Machine Learning**”, Andrew Ng

[**https://see.stanford.edu/materials/aimlcs229/cs229-notes1.pdf**](https://see.stanford.edu/materials/aimlcs229/cs229-notes1.pdf)

[4] “**Parallel Spectral Clustering in Distributed Systems**”,Wen-Yen Chen,Yangqiu Song, Hongjie Bai,Chih-Jen Lin,Edward Y. Chang

[**https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5444877**](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5444877)

[5] **“Code”,** [**https://github.com/cseuconn/5905-fall-2021**](https://github.com/cseuconn/5905-fall-2021)