# CSE 511 - Project 2 Report

Ankush Mishra - aam6386@psu.edu
Vinay Vemuri - vvv5079@psu.edu

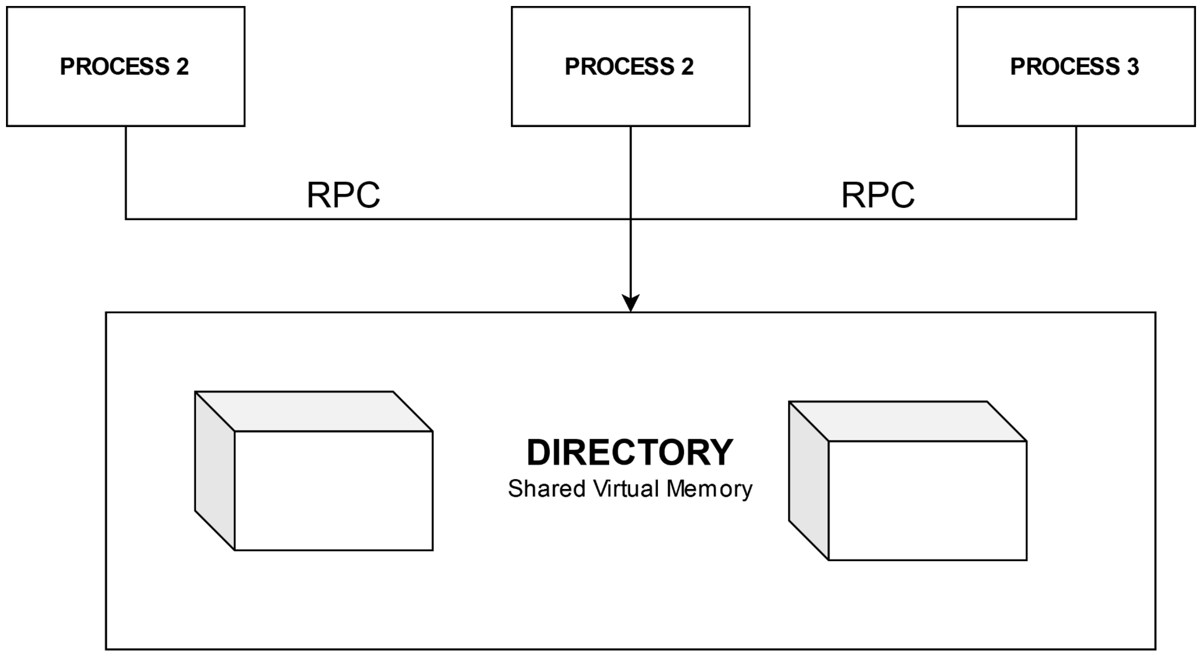
## Group Members:

Ankush Aniket Mishra (aam6386@psu.edu )
Vinay Kumar Vemuri (vvv5079@psu.edu)

## Introduction

This report serves as a brief summary of the approach to the project, the lessons learned, and how the work was distributed among the both of us. The goal of this project was to start a program/function on multiple machines, such that the application memory is shared across multiple nodes.

## Application Design



**CSE 511 - Project 2 Report**

Ankush Mishra - aam6386@psu.edu
Vinay Vemuri - vvv5079@psu.edu

The above diagram serves as a high level design of how the application can be used and is implemented.

## Overall Logic Implementation:

### Part 1:

We implemented the Ricart-Agarwala Distributed mutual exclusion locking mechanism based on the original implementation from the paper. The algorithm is provided in the course slides and in the paper, which essentially uses a sequence number and highest sequence number based on lamport clocks to implement the functionality. Each lock request is expected to be replied by all the nodes for a single node to get access. Until then the individual request is blocked. This served as the basis for our architecture of the application and how to structure it.

### Part 2:

The implementation of DSM fairly followed from what we learned from the lecture. The directory node had the table stored per memory shared( malloc or register data segment). Please note that in our implementation we do not allow users to name a memory region from malloc with “default”. The columns of the table indicate the number of nodes in the network and the number of rows indicate the number of pages for that shared memory region. Each row has a lock associated with it. Further client initiates and requests accesses through rpc calls, and directory grants or revokes page permissions through rpc calls. Depending on the circumstance, the actual latest copy of the page data from a node may piggy back in the reply messages of the rpcs. One of the most important functions of this part was mprotect as it

Ankush Mishra - aam6386@psu.edu
Vinay Vemuri - vvv5079@psu.edu

allowed us to change the permissions of pages. Lastly, this entire part would kick off from pages that don't have the right permissions, would have a segmentation fault. The sig handler deciphers whether it's a write or read fault and our DSM implementation follows accordingly. One last point to mention is that we assume the size provided for memory allocation or registering data segment would be a multiple of 4096.

One catch of this implementation was that we needed to use posix_memalign instead of malloc for allocating page aligned memory.

## Part 3:

MapReduce based Word count is a fairly common problem in distributed architectures, we used a globally shared array containing the individual word frequencies. To do so each node would be allocated some chunk of the file to process and the results would be then merged during the reduce function onto the global shared array. The use of the DSM shared segment came in handy in this scenario.

## Part 4:

Kmeans was pretty straightforward as it is an application on top of map reduce, similar to word counter. We initially register a distributed shared memory that contains the Euclidean distance of the points that the nodes calculated a subset of and then calculated and grouped based on the closest centroid of the 4. This part is completely in the map phase and the task is shared among the nodes in the distribution. In the reduce phase, each node is responsible for taking at least one centroid and calculates the average of the points to give the new centroid.

**CSE 511 - Project 2 Report**

Ankush Mishra - aam6386@psu.edu
Vinay Vemuri - vvv5079@psu.edu

## Difficulties Faced:

The project is without a doubt bigger and more challenging than the previous project. The biggest challenge we probably faced was designing and figuring out the dsm, rpcs, and locks and how they all integrate with each other.

Debugging across different machines especially how data is transferred, whether the relevant data is transferred, permissions, locks etc was challenging to get working right.

## Special cases project does not work for:

From our builds and tests, all required components of the project work as intended. We have in fact added our own extra test cases and work to ensure everything is working as expected.

## Extra work done beyond the requirements:

Additional test cases for DSM Malloc were created, various race conditions and other work has been validated and ensured that no intermittent data corruption can occur in our implementation.

## Distribution of work:

The distribution of work between the team members was equal.

1. The RPC Design, DSM Malloc and Ricart Agrawala and map reduce was done by Ankush.
2. While DSM DataSegment and KMeans was shared with Vinay.
3. The tasks of debugging, testing, and designing were split equally.

# CSE 511 - Project 2 Report

Ankush Mishra - aam6386@psu.edu
Vinay Vemuri - vvv5079@psu.edu

## **References:**

Glenn Ricart and Ashok K. Agrawala. 1981. An optimal algorithm for mutual exclusion in computer networks. Commun. ACM 24, 1 (Jan. 1981), 9–17. DOI:https://doi.org/10.1145/358527.358537

C++ Reference: https://en.cppreference.com/w/